1-15-2019

An Airlifted Tidal Mesocosm for Oil Degradation Studies

Daniel Christopher Alt
Louisiana State University and Agricultural and Mechanical College, dalt1@lsu.edu

AN AIRLIFTED TIDAL MESOCOSM FOR OIL DEGRADATION STUDIES

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

The Department of Civil and Environmental Engineering

by
Daniel Christopher Alt
B.S., Louisiana State University, 2010
M.S., Louisiana State University, 2015
May 2019

ACKNOWLEDGEMENTS

This dissertation has taken a long time to come to fruition, and I would like to thank many people for helping me through it. Foremost I would like to thank my parents, Patricia and Bill Alt, for helping me through graduate school, in editing this document, and having patience while I finished. This dissertation was made possible by a grant from The Gulf of Mexico Research Initiative to the Coastal Waters Consortium. The funders had no role in the design, execution, or analyses of this project. I would also like to thank members of my committee for guidance: Dr. Brian Roberts, Dr. Jill Trepanier, Dr. Celalettin Ozdemir, Dr. Chandra Theegala, and most significantly, Dr. Ronald Malone. Dr. Malone has helped me through a few topic changes, and innumerable rewrites. I am also in debt to his wife, Ms. Sandra Malone, who has also encouraged me throughout this process and has tolerated many late discussions with Dr. Malone. There are numerous other people who have helped me with this document and have given me the ability to finish. Dr. Chad Cristina helped me immeasurably by improving my writing and helping me understand what it means to be an engineer, and Lisa Weaver was a great friend I could rely on throughout this process. I am also in my fellow graduate students' debt, as they have given me the encouragement I needed to complete this document, and I would like to especially thank Rebecca Tabor, Sima Hannani, Fatemeh Fahandezh, Dr. Marlon Greensword, and Dr. Doorce Batubara. I would also like to thank several programmers who took the time to write libraries that were used in the various programs that make this dissertation possible: Tom Igoe, Andrew Rapp, Mathijs Kooijam, zoomkat, Kevin Kuwata, Daniel Shiffman, and Robin2. Finally, I would like to thank the staff at LUMCON who aided me in working on the mesocosms: Dr. Charles Schutte, Dr. Ryann Rossi, Caleb Bourgeois, Brendan Kelly, Ron Scheuermann, Ekaterina Bulygina, and Logan McPherson. Thank you all for everything, this dissertation would not be possible without you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

TERMS AND ABBREVIATIONS USED

AOTR	Actual Oxygen Transfer Rate
API	Application Programming Interface
ACTR	Actual Carbon Dioxide Transfer Rate
CSV	Comma Separated Value
CTR	Carbon Dioxide Transfer Rate
CWC	Coastal Water Consortium
C_{Sat}^{C}	Dissolved Carbon Dioxide Saturation Concentration (mg L^{-1})
$(C_{Sat}^{C})_{20}$	Dissolved Carbon Dioxide Saturation Concentration at 20°C (0.5 mg L^{-1})
C_{m}^{C}	Standard Measured Concentration of Carbon Dioxide (taken as 1 mg L^{-1})
C_{in}^{C}	Dissolved Carbon Dioxide Concentration Coming into the Airlift (mg L^{-1})
C_{out}^{C}	Dissolved Carbon Dioxide Concentration Coming out of the Airlift (mg L^{-1})
C_{Sat}^{O}	Dissolved Oxygen Saturation Concentration (mg L^{-1})
$(C_{Sat}^{O})_{20}$	Dissolved Oxygen Saturation Concentration at 20°C (taken as 9.07 mg L^{-1})
C_{m}^{O}	Standard Measured Concentration of Oxygen (assumed to be 0 mg L^{-1})
C_{in}^{O}	Dissolved Oxygen Concentration Coming into the Airlift (mg L^{-1})
C_{out}^{O}	Dissolved Oxygen Concentration Coming out of the Airlift (mg L^{-1})
GUI	Graphical User Interface
IDE	Integrated Development Environment
$K_{C}^{20.3}$	Constant Taken from the Slope of the CTR vs. Time (unitless)
$K_{O}^{20.3}$	Constant Taken from the Slope of the OTR vs. Time (unitless)
$(K_L a_C)_{20}$	Gas Transfer Coefficient for Carbon Dioxide at 20°C (L min^{-1})

$(K_L a_O)_{20}$	Gas Transfer Coefficient for Oxygen at 20°C (L min^{-1})
kW_{ad}	Adiabatic Power (kW)
L	Lift
LUMCON	Louisiana Universities Marine Consortium
OTR	Oxygen Transfer Rate
P_1	Blower Inlet Pressure (kPa)
P_2	Blower Outlet Pressure (kPa)
Q_1	Volumetric Flow Rate (m^3 ft)
Q_G	Gas Flow Rate (L day^{-1})
Q_L	Water Flow Rate (L day^{-1})
S	Submergence
SAE	Standard Aeration Efficiency
SCTR	Standard Carbon Dioxide Transfer Rate
SLDM	Static Low-Density Media
SOTR	Standard Oxygen Transfer Rate
SSE	Standard Stripping Efficiency
V	Volume of Water Being Degassed (L)

ABSTRACT

Mesocosms were constructed to allow scientists to isolate variables in a microtidal marsh environment, mimicking the natural conditions found in Terrebonne Bay. The control offered by these mesocosms is given by a process control system that takes user inputs and automates water movement. Twelve mesocosms were constructed, each holding 4.02 m^3 (142 ft^3) of marsh soil and 3.88 m^3 (137 ft^3) of water to give 3 experimental levels and a control, all in triplicate.

Each mesocosms that was constructed contained marsh plants in soil plugs 9 feet in diameter and 3 feet deep. These marsh plugs are held in fiberglass tanks that allow for tides to vary up to 11 inches above and below the marsh surface, with the low tide dropping along an exposed side of the marsh. The backbone of the mesocosms' design is the airlift, which allows for centralized air blowers to move water to mimic tides, keeping each system isolated and mechanical parts shielded from corrosion. The system exchanges experimental water with water from the nearby bayou using a system of ball valves and pumps and returns used water to the bayou after going through a treatment system.

The operator of the system uses a GUI (graphical user interface) to input commands to a computer program that then operates the various components of the system. The central program was coded using the Processing© program with control over various valves and sensors in the field controlled with the Arduino Uno. The two programs communicate wirelessly using the XBee software and hardware. The results of the process control system show that the system closely matches what was input by the user, giving the operators of the system the ability to control the system without needing a deep understanding of computer code.

CHAPTER 1. INTRODUCTION

1.1. INTRODUCTION

The marsh environment, like all ecosystems, has several interdependencies that makes the study of any specific area challenging. Studies are usually broken down into small scale microcosms or large-scale field studies. Microcosms allow for a great deal of control but run into issues scaling. Field experiments allow the study of what is going on in the environment but give little control and can often have unexpected variables or events interfere with the experiment. Mesocosms give a balance between microcosms and field studies as they give the ability for operators of the experiment to have a great deal of control while being large enough that the mimic natural conditions.

The mesocosm experiments called for in this experiment initially came out of the need to study the effects of the Deepwater Horizon Oil Spill. Previously, these studies ran into issues isolating the effects of the spill from the interaction of various forces in the marsh. The planned mesocosms would give the ability to create experimental designs that could be set up to isolate factors. This is accomplished by creating a process control system that mimics natural conditions based on user inputs.

1.2. BACKGROUND

The mesocosm is an experimental design that replicate natural conditions while giving some degree of control and whose size is loosely defined as one to several thousand liters sized (Stewart et al., 2013). The mimicking of natural conditions allows the operator of the system to conduct experiments with more interactions but forces a more robust system design to contain experiments. The costs associated with larger and more complex experiments has made mesocosms cost prohibitive until recently. The availability of not only hardware to operate the

processes of large experiments, but the software to control the hardware has rapidly become available in the past few years.

In 2014, researchers at the CWC identified that mesocosms were needed to continue to study the effects of oil spills in the marsh. Previous studies on oil included the analyzation of oil distribution in the marsh (Ramsey et al., 2014), its environmental impacts (Beyer et al., 2016), and the persistence of oil in the marsh (Turner et al., 2014). However, efforts to continue to study oil in the marsh is increasingly difficult due to multiple stressors affecting the marsh systems and influence by physical forcings as shown in Figure 1.1. This is further complicated with concerns on the length of time since the oil spill, the difficulty isolating impacts against variability of marsh sites, and the heavily oiled sites being likely to have been lost after exposure (Rabalais et al., 2014).

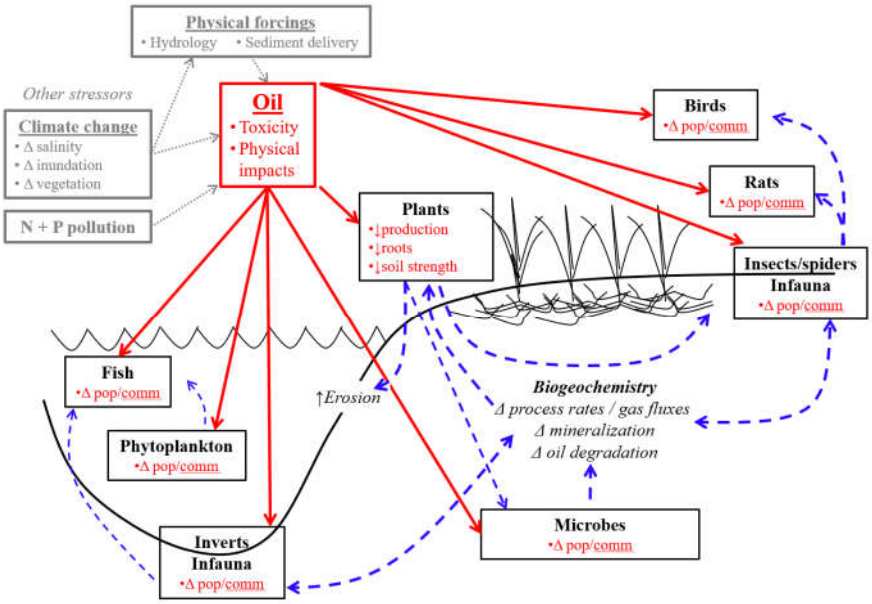

Figure 1.1. Direct and indirect impacts of oiling on marsh ecosystems (Rabalais et al., 2014). This is a schematic illustrating the direct (solid red lines) and indirect (blue dashed lines) impacts of oiling on marsh ecosystems

The mesocosms that were called for by the CWC would allow for the isolation of different variables while the interaction of oil and the marsh are studied. Elmgren and Frithsen (1982) and Dellagnezze et al. (2016) used mesocosms to study the effects of oil breaking down in a saltwater column and used water pumps to circulate the water. Lin and Mendelssohn (2012) and Judy et al. (2014) used marsh mesocosms to study oil's effect on plants in experiments that collected water that had drained through the soil and reapplied it to the marsh surface. Pennington et al. (2004) and Batubara (2014) used mesocosms to study the breakdown of chemicals with some tidal control with soil in tiered sections. Pennington et al., (2004) used a water pump while Batubara (2014) used an air pump and air chamber to displace water. Notably, Batubara (2014) was able to use a single air pump to control four mesocosms while Pennington et al. (2004) used twelve different pumps to move water in twelve individual mesocosms.

1.3. MESOCOSM OVERVIEW

The design that this dissertation focuses on involves the creation of 12 mesocosms that contain plants and soil that experience daily tidal amplitudes typical of coastal LA. These mesocosms would allow for 3 experimental levels and a control, all in triplicate. The mesocosm core components are two fiberglass tanks, one to hold the marsh (marsh tank), and the other to hold the water used in water movement (surge tank). The surge tanks will be approximately 300 gallons in volume and allow for the removal and addition of water via a series of control valves. Trapezoid-shaped tidal cycles will be created by operators to regulate: (i) the rate of the water level rise, and (ii) how long the water remains at a plateau and (iii) the rate of decline. A drawing of the two tanks is shown in Figure 1.2.

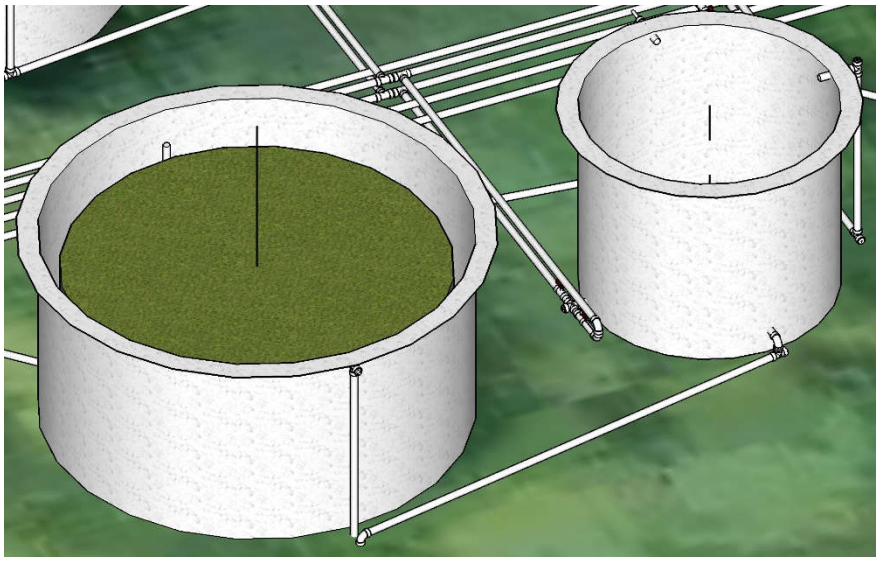

Figure 1.2. Marsh tank (left) and surge tank (right)

Airlifts will be used to move water to simulate daily tides according to design guidelines initially set by Loyless and Malone (1998), and further refined by practice (Gudipati, 2005; Johnson, 2008; Hearn, 2009; and Alt, 2015). Studies involving the airlift have focused on its ability to aerate and degas while moving water in low head operations. One of the first studies, Loyless and Malone (1998), looked at the aeration and degassing behavior of airlifts with different methods of air delivery and found that for low head applications, the water delivery demands would be less than the gas exchange requirements. Gudipati (2005) would take these studies along with some design experience and developed rules of thumb for the sizing and installation of airlifts for water flow in recirculating applications (Malone and Gudipati, 2005). Johnson (2008) next used the airlift in line with a static low-density media (SLDM) filter to treat domestic wastewater. Hearn (2009) further studied the aeration and degassing kinetics using airlifts to support a Recirculating Aquaculture System (RAS) with adult Yellowtail broodstock (*Seriola lalandi*). Finally, Alt (2015) took these studies to create a computer model that simulated how to increase the yield of a traditional RAS by modeling management strategies: cross flow (water circulation between tanks) and cohort stocking.

To mimic natural water quality conditions, a system to bring water to the mesocosms and a system to treat water after it runs through the mesocosms is needed (Figure 1.3.). The exchange of water is handled through a process control system with Arduinos, float valves, holding tanks, and ball valves. All mesocosm's discharge water will be kept in a large detention tank that will eventually treat the water through a combination of biofilters, carbon filters, and UV lights to degrade any oil-related products.

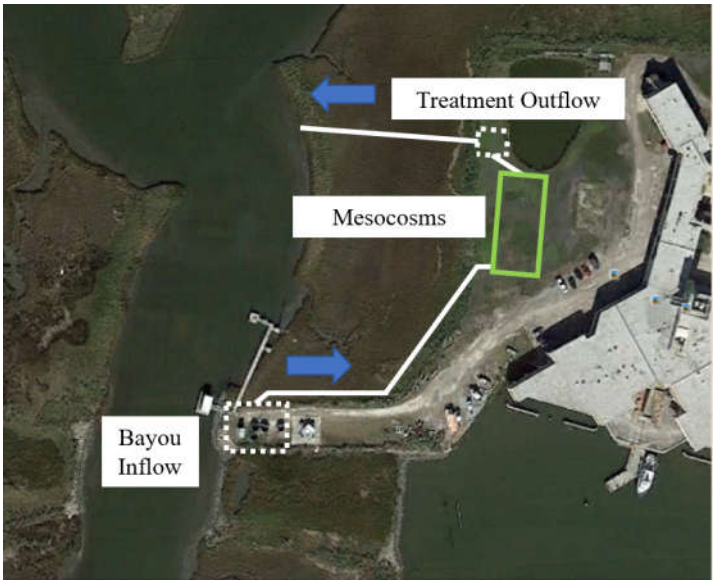

Figure 1.3. Areas of mesocosm system

An important feature of the system is its ability to control water levels through simulated tides that is regulated by the process control system. This is set up using the Processing© and Arduino software and hardware platforms that would control the water flow mainly using airlifts. The Processing© program is an open source computer program based on Java that created in 2001 by Ben Fry and Casey Reas primarily for use in the visual arts (Processing©, 2018), with the Arduino Integrated Development Environment (IDE) based off it (Arduino, 2018). The Arduino line of microprocessors is a relatively new open source technology with the company started in 2005 (Arduino, 2017), and the microprocessor used here, the Arduino Uno, being

introduced in 2010 (Banzi, 2010). The airlift is much older, with earlier use focusing on the mining, oil, and aquaculture industries to move liquids starting as early as 1797 (Castro et al., 1975; Gudipati, 2005).

The Arduino technology has been used in several experiments used to control the movement of water. Ferrarezi et al. (2015) used the technology to build an irrigation system that also monitored soil levels. Miller and Long (2015) used the Arduino to change the tidal pattern in laboratory mesocosms using a drain pipe that would rise and fall to control the water levels of tidal patterns. While it was not used to control tides, Lockridge et al. (2016) used the Arduino to control sensors for coastal water monitoring in a coastal bay environment. Finally, Lee et al. (2016) came the closest to the design used in the mesocosms as the Arduino was used with solenoid valves in a mesocosm looking at saltwater intrusion in a tidal freshwater wetland mesocosm, but the water flow was controlled by gravity.

1.4. OBJECTIVE AND SCOPE

This dissertation documents the design and validation of the mesocosms, focusing on the physical layout and the process control system. A view of the mesocosms and water motion that the system must regulate are shown in Figure 1.4., with the expected water movement show in Figure 1.5.

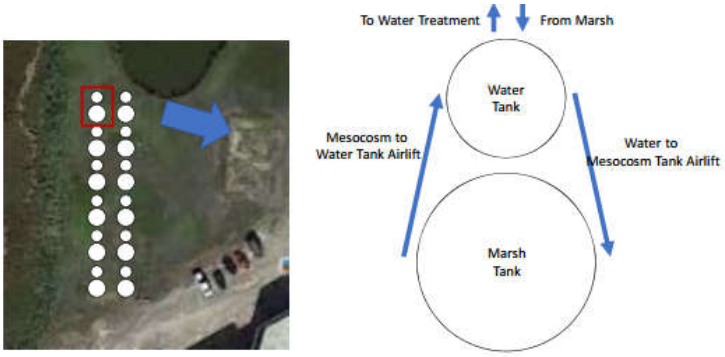

Figure 1.4. Location of mesocosms in the field along with the 4 ways in which water is moved.

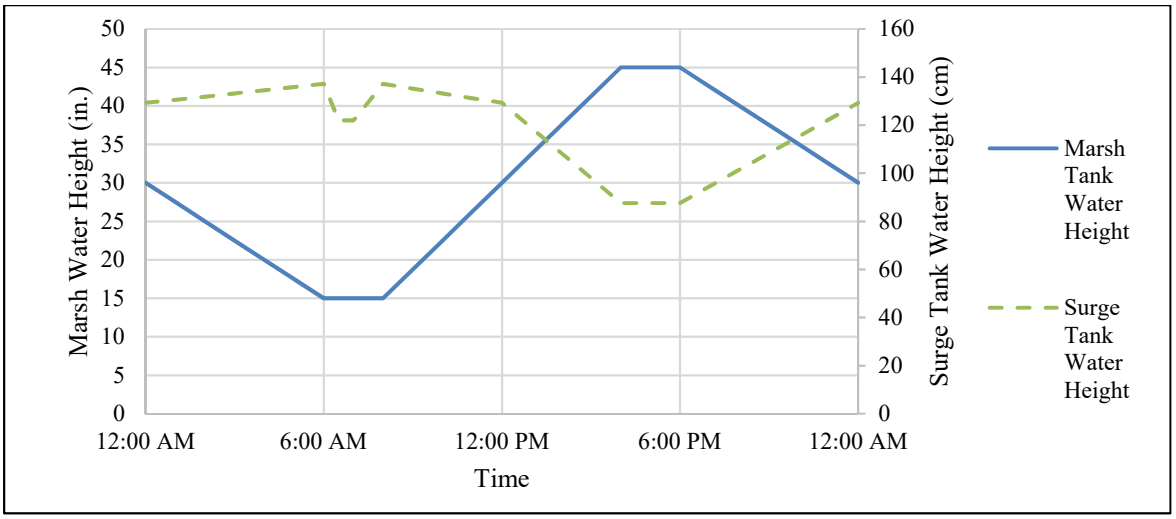

Figure 1.5. Water heights in both the marsh and surge water tanks

The second chapter of the dissertation will lay out the history of airlifts and the design rules from experience in aquaculture. It summarizes the work of a literature review survey, including 4 Master's Thesis from Louisiana State University (LSU), drawing data from both experiments and computer models (Gudipati, 2005; Johnson, 2008; Hearn, 2009; and Alt, 2015). While the airlift has been used extensively in aquaculture, this chapter will lay out design guidelines for water movement that have driven the design of the mesocosm.

In Chapter 3, the physical design and operation of the mesocosms will be explained along with the rationale behind the water movement and how it is managed by a process control system. This system includes a central process control program, a wireless communication network, and several microprocessors in the field that are used to collect data and operate the valves allowing water movement. The validation of the mesocosms will focus on the water height from the field vs. the programmed tides in addition to the water exchange with the surrounding bayou. The conclusion of the dissertation will wrap up the design of the mesocosm and give recommendation for future work.

CHAPTER 2. AIRLIFT DESIGN AND OPERATION

2.1. INTRODUCTION

In the past 50 years, the production from aquaculture has grown to meet the growing demand for seafood from an increasing population growth as the wild catch industry has seen its numbers stagnate. From the 1960's to 2013, not only did the world population double, the worldwide per capita apparent fish consumption increased from an average of 9.9 kg to 19.7 kg, creating a fourfold increase in demand (FAO, 2014). In the past 25 years, the main increase in fish production has mainly come from aquaculture rather than wild catch (FAO, 2014). As this industry has grown, it is facing challenges as more intensive practices have led to the degradation of water quality, limited water and land resources, and the occurrence of disease outbreak. These concerns have led to the development of new technologies in recirculating aquaculture that eliminate most of these issues (Gutierez-Wing and Malone 2006).

Recirculating Aquaculture System (RAS) is a facet of aquaculture developed to allow the culture of fish in tank systems. These systems are designed to maintain adequate degassing, aeration, circulation, nitrification and solids removal with minimal water discharge. Designs of RAS have shifted towards more efficient designs due to competition from ponds and other forms of aquaculture. The use of an airlift allows for the consolidation of various processes into one unit as the airlift can circulate water and aid in aeration and degassing, lowering energy and capital costs. The mechanism of the airlift involves the density differences created when air is injected into a column of water, airlifts are a useful low lift pumping alternative.

As the airlift became increasingly used, airlifted RAS design rules on its processes were created. These rules were based on empirical observations as the design was refined over time. Originally introduced by Malone and Gudipati in 2005, the different factors of design were

consolidated into “Interim” design guidelines developed from laboratory and commercial operation (Malone and Gudipati, 2005; Malone and Gudipati 2007) for the airlift/floating bead filter combination.

2.2. BACKGROUND

The airlift has been used extensively in the oil and mining industries to move liquids before being used in aquaculture to move water and maintain dissolved gasses. The airlift concept was first discovered by Carl E. Loescher in 1797 and was initially used mainly in the mining industry and to pump wells and would find a use in sewage treatment a century later (Castro et al., 1975). The airlift would also be used in aquaculture due to its ability to simultaneously circulate, aerate, and degas water (Loyless and Malone, 1998; Gudipati, 2005; Castro, 1975; Reinemann et al., 2001).

The design of the airlifts used in aquaculture differed from those in mining by widening the pipe diameter and decreasing the length of pipe, partially as the air pumps of the time being unable to inject air at deep pressures (152cm or 60 inches) because of the threat of gas supersaturation. This can occur when air bubbles are under water pressure greater than atmospheric pressure, which allows gases to transfer to the water at greater concentrations than normal. The result is chronic or acute diseases in aquatic species, such as gas bubble disease, where the supersaturated dissolved gas is forced out of the water and forms bubbles within the fish. The depth that this occurs will depend on the temperature, salinity, depth of injection, and type of aeration device. In an airlift, it is usually avoided if submergence is kept below 12 feet (Huguenin and Colt, 1992; Wheaton, 1977).

The airlift’s two fundamental components are an air delivery/injection tube and a water discharge pipe or draft tube shown in Figure 2.1. (Castro and Zielinski, 1980; Parker, 1981).

These components are usually made of PVC pipe, though the cross-sectional area does not matter so long as general rules of thumb are followed. The airlift operates as the air injection lowers the water density in the draft tube, allowing the denser tank water to push it upwards towards the headpiece (Wheaton, 1977). While the two-phase air-water mixture travels through the pipe, gas exchange occurs for oxygen and carbon dioxide present (Reinermann and Timmons, 1989; Loyless and Malone, 1998; Hird et al., 2000).

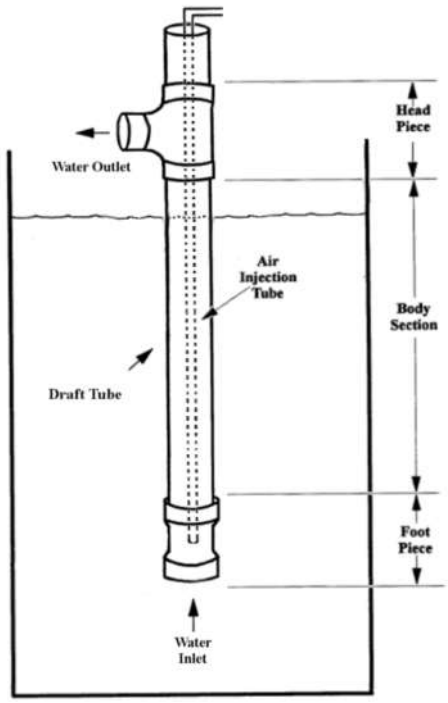

Figure 2.1. Illustration of an airlift's components (modified from Loyless, 1995)

Nicklin (1963) identified the gas flow, the depth of air injection, the distance water is lifted to, and the pipe size as key factors impacting airlift performance. The depth of the air injection is defined as the submergence (S) as shown in Figure 2.2.(a). The submergence is a key factor defining the net buoyant force. This consequently raises the rate of the water delivery and the possible height of the headpiece outlet or lift (L). Given a set S and L, the flow rate of air injected (G) will determines how much water flows (Q). When in operation, the headloss

increases as the frictional losses associated with pipes and fittings climb, increasing the lift while simultaneously decreasing the submergence dynamically (Figure 2.2. (b)). In some cases, the lift is usually equal to the dynamic headloss as shown in Figure 2.2. (c). In this set up, the draft tube is split into the draft tube that contains the injected air and the approach pipe in which water is routed to the airlift.

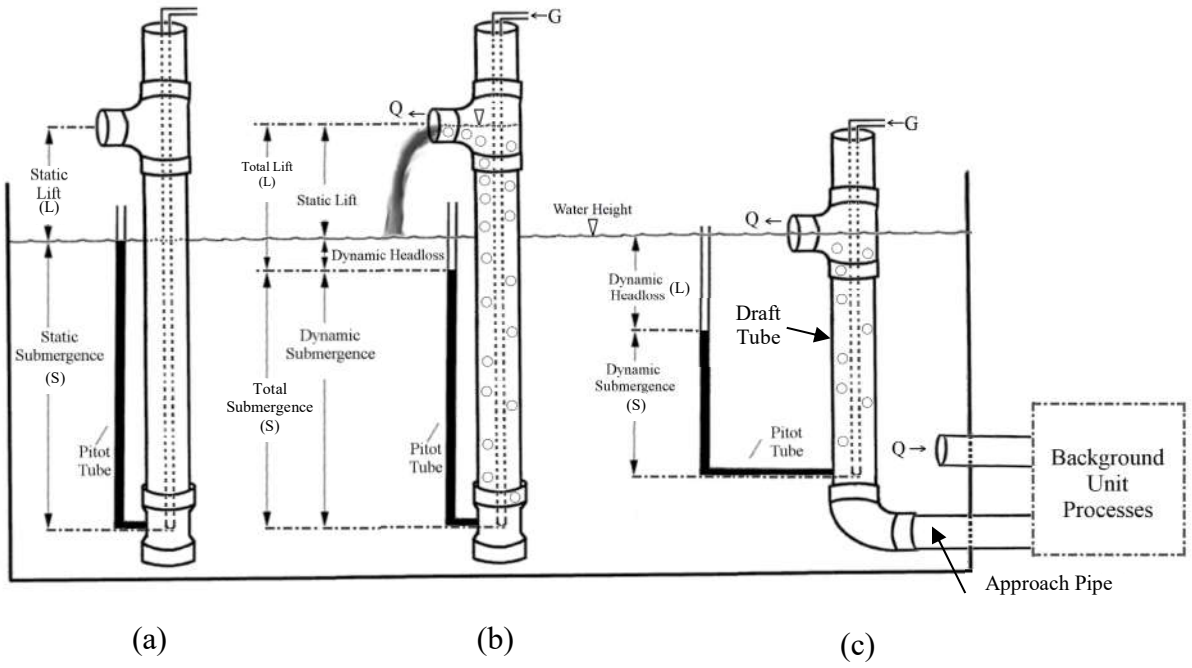

Figure 2.2. No flow (a), lift (b), and operational (c) conditions (modified from Loyless, 1995)

The performance and ensuing gas transfer of an airlift are impacted by bubble size characteristics (Wheaton, 1977). The bubble pattern is a result of the amount and methodology of air injection, with the three most common shown in Figure 2.3: bubble (a), bubbly-slug (b), and slug flow (c), and are a result of the (Barnea and Taitel, 1986). The bubble flow pattern is usually a result of low rates of air injection or diffused types of injection devices (air- stones), with larger amounts of air injection and open-ended injection ports resulting in bubbly-slug and slug flows, the two most commonly found in airlifts (Parker and Suttle, 1987; Hearn, 2009).

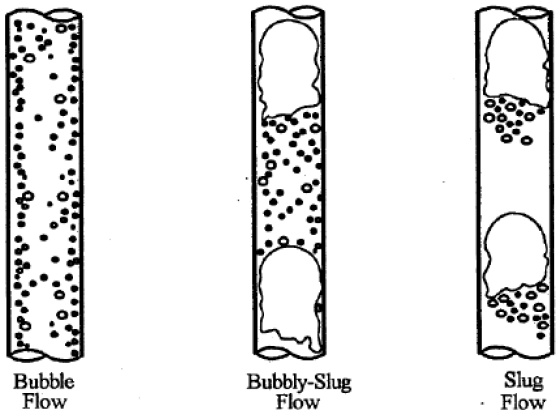

Figure 2.3. Different bubble sizes seen in an airlift (Hearn, 2009)

The bubble characteristics has an impact on the airlift performance. In the bubble flow regime, gas transfer is greater, but it may not allow for the desired liquid flow rate to be achieved. Open-ended air injection devices have a lower amount of gas transfer, but allow selected air pumps to maximize water flow, are resistant to calcification, and thus, are usually preferred in commercial airlifts (Reinemann and Timmons, 1989; Loyless and Malone, 1998; Malone and Gudipati, 2007).

The airlift provides a simpler, more compact, and energy efficient device for low-head water circulation compared to technologies like the centrifugal pump (Spotte, 1970). By using air to move water, the airlift does several processes at once: water movement, oxygenation and stripping carbon dioxide (Hird, 2005). Within low head operations, the airlift is more cost efficient compared to a centrifugal pump with lower operation and maintenance costs due to the absence of moving parts and minimal required technical expertise (Hearn, 2009). The energy required for an airlift to circulate and aerate were close to one-third the cost of a centrifugal pump and aeration system (Reinemann, 1987), moving the highest volume of water per unit of

energy if designed properly (Timmons et al., 2001; Hearn, 2009). The increase in efficiency of this consolidation overcomes any efficiency loss as individual processes is not fully optimized (Bellelo, 2006). In addition, the blowers used to power the airlifts can be installed away from the tanks holding fish, minimizing the corrosion and electrical issues associated with a wet environment.

2.3. DESIGN RULES EVOLUTION

Studies on the airlift have focused on its ability to aerate and degas while moving water. Loyless and Malone (1998) looked at the aeration and degassing behavior using different methods of air delivery in airlifts and found that for low head applications, the water delivery demands would be less than the gas exchange requirements. They observed that aeration in airlifts was less than the same delivery device in open water. It was concluded that airlifts should be sized for water movement with the remaining aeration demand moved to in-tank devices. This approach offsets the aeration and carbon dioxide stripping needed, lowering the overall cost of water movement. It was also found that the requirements for aeration would surpass the carbon dioxide stripping needs in a freshwater system, eliminating the need to size a system for degassing if mixed air is used.

Gudipati (2005) developed several rules of thumb for the sizing and installation of airlifts for water flow in recirculating applications (Malone and Gudipati, 2005). The focus of Gudipati's thesis was the design of a soft-shell crawfish recirculating aquaculture system, specifically looking at the gas to liquid ratio, and the lift distances. Gudipati's experiments concluded in several design rules of thumb for their system that could be applied to a more general application.

Johnson (2008) next used the airlift in line with a static low density media (SLDM) filter to treat domestic wastewater. Johnson's focus was on the submergence to lift ratio and air injection rate that would allow the SLDM filter to achieve optimal treatment of BOD and nitrogen. Johnson's experiments focused on a 6-inch airlift connected to a 25 ft^3 static low density media filter, but the data gathered would help to refine the design criteria first proposed by Gudipati.

Hearn (2009) took the design rules laid out and used them to design an RAS supporting adult Yellowtail broodstock (*Seriola lalandi*) in a marine RAS. The experiments carried out looked at the gas exchange in an airlift with varying diameter, gas to liquid ratios, and lift to submergence ratios at different loading rates with supplemental aeration needed at extensive operations. The gas exchange studies focused on empirically finding the oxygen addition and carbon dioxide stripping rates. Once these were found, Hearn created a computer model to evaluate how the airlift would behave under different feeding regimes.

Finally, Alt et al. (2010) and Alt (2015) created a series of computer models to estimate how much the production of a RAS would increase with various load leveling stocking strategies. Thanks to the data gathered by previous researchers, the prediction of how an airlift affects ammonia, oxygen and carbon dioxide could be made. With these predictions, it became possible to reconfigure the layout of a facility to take advantage of unused capacity and find ways to increase the production of a system.

2.4. DESIGN RULES RATIONALE FOR WATER FLOW

2.4.1. WATER FLOW

Airlift do not operate well if they must overcome a large amount of energy. Energy is primarily lost due to an increase in elevation or due to friction from the wall of the pipes and can

be limited by keeping the water velocity and elevation change low. Due to this and lower gas transfer kinetics compared to open water systems, Loyless and Malone (1998) recommended that the airlift be primarily sized for water circulation. As such, all aspects of the design will revolve around water flow with savings from degassing and aeration calculated later.

2.4.2. LIFT HEIGHT (L) AND SUBMERGENCE (S) DEPTH

The submergence and lift rations have been used with the diameter of the pipe to try and predict the water flow for a given air injection in Castro and Zielinski (1980) and Parker and Suttle (1987). These studies showed that an increase in the vertical lift height had a greater effect on water flow rates in larger diameter pipes compared to smaller pipes (Hearn 2009). The water flow increased with a constant lift and increased submergence depth, with studies on airlifts showing the most efficient ratio ranges from 3:1 to 4:1 (Wheaton, 1977), 4:1 (Timmons et al., 2001), as well as 4:1 to 5:1 (Malone and Gudipati, 2007).

As the S:L ratio increases, the optimal gas to liquid ratio continues to improve as shown in Figure 2.4. (Johnson, 2008). The design ratio of a 4:1 submergence to lift ratio was found to be an optimal design ratio. As the S:L ratio increases from 3:1 to 4:1, the optimal gas to liquid ratio falls from 2:1 to 1.3:1, allowing for more water to be pumped with less air. If the design ratio falls below 2:1, the water flow reaches a plateau where increasing the air flow achieves little. Increasing the submergence to lift past a 4:1 ratio with a higher lift raises issues with tank construction costs and risk of gas supersaturation disease on fish.

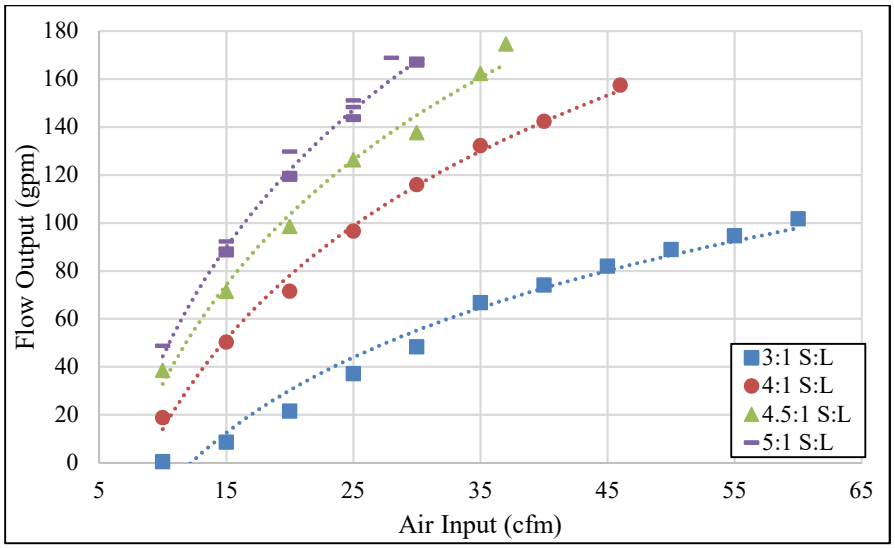

Figure 2.4. Behavior of a set 12" total lift given different S:L ratios (Johnson, 2008)

2.4.3. LIFT PIPE DIAMETER

In sizing the piping for an airlift, the velocity of water going through the pipe controls several factors, including scour, headloss, and solids deposition. Malone and Gudipati (2005) made recommendations for sizing airlifts in sequence with PolyGeyser® floating bead bioclarifiers, citing velocity requirements at 0.31 m s^{-1} (1 ft s^{-1}), which has since risen to 0.47 m s^{-1} (1.5 ft s^{-1}) (Hearn, 2009; Malone, 2017).

The water velocity is set at 2-3 ft s^{-1} in the approach pipe to prevent solids deposition while preventing pipe erosion. Higher velocities (> 5ft s^{-1}) can prevent biofouling but can lead to the erosion of the pipe (15 ft s^{-1}) and lead to water hammer becoming a design issue (Huguenin and Colt 1992). Low velocities (<3 ft s^{-1}) keep headloss low, ensuring the design is energy efficient. As shown in Figure 2.5., headloss due to friction becomes an issue as the velocity rises above 3 ft s^{-1} and can approach several feet per 100 feet without including other causes of headloss such as fittings.

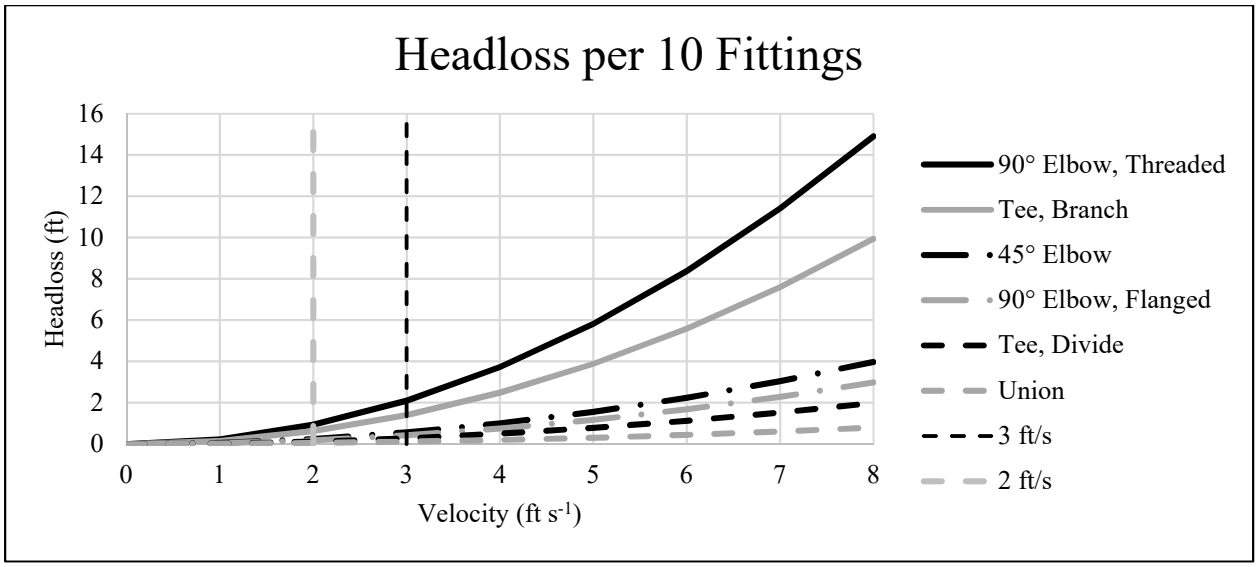

Figure 2.5. Headloss in various pipes (Gudipati 2005)

Pipe diameter also contributes to the overall airlift performance with larger pipes producing greater liquid flows (Parker and Suttle 1987, Hearn 2009). Data from Johnson (2008) shows how increasing the pipe diameter from 2 inches to 12 inches increases Q_L by a factor of 10 assuming a logarithmic relationship (Wurts, 1994). Larger pipes also allow for increased interaction between air/water contact times enhancing gas transfer (Hearn, 2009). However, the larger the pipe is, the more air is needed to initially move the water, making them inefficient for low flow situations.

2.4.4. AIR DELIVERY

The gas to liquid flow ratio (Q_G:Q_L) determines the level of efficiency that a pump operates under (Reinemann and Timmons, 1989; Wheaton, 1977). As air injection (Q_G) exponentially increases, the resulting fluid discharge (Q_L) will linearly increase until some optimal or peak air injection is met, at which water flow will slowly decline (Todoroki et al., 1973; Castro and Zielinski, 1980; Parker and Suttle, 1987; Wurts et al., 1994; Reinemann et al., 2001; and Hearn, 2009).

Malone and Gudipati (2005) developed a Q_G:Q_L less than two to assure energy efficiency in aquaculture operations, experience with designs in the field allowed for the Q_G:Q_L ratio to be adjusted to ~2 currently (Hearn, 2009). Johnson (2008) found that as Q_G:Q_L ratio was closely tied to the S:L ratio as shown in Figure 2.6. with a 6-inch airlift pump. As the submergence to lift ratio increases, the Q_G:Q_L will continue to decrease if the water flow is kept above a certain threshold.

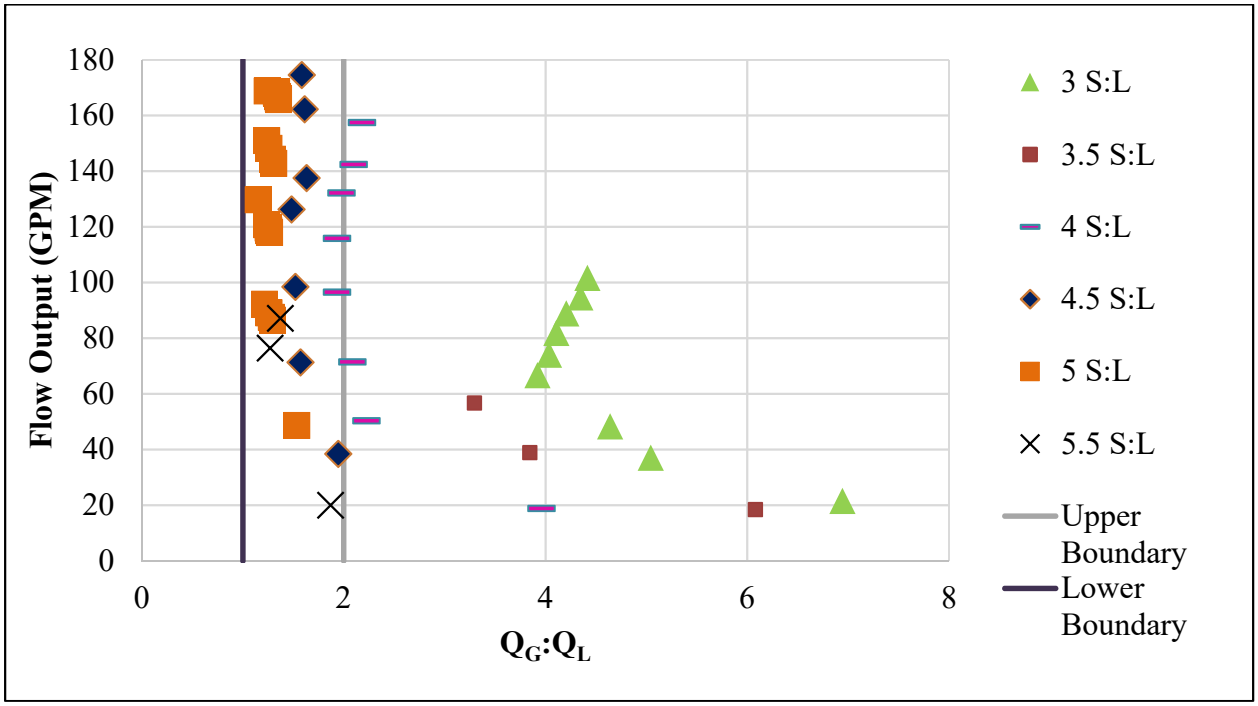

Figure 2.6. Relationship between Q_G:Q_L and S:L ratios from Johnson (2008)

2.5. GAS TRANSFER

The support of dissolved gas stabilization at concentrations allowing for optimal fish and bacterial growth is one of the critical roles that an airlift plays in a RAS facility. Dissolved oxygen is most often the limiting factor in aquaculture facilities, with interruptions in operation causing fish kills in minutes. However, as fish densities and feed rates have increased, CO_2 management has become critical to a successful unit and is complicated by the carbonate in the water which interacts with dissolved carbon dioxide (Hearn, 2009). In an airlift, the degassing

and aeration occur simultaneously as the air and water mix with more air resulting in higher rates of each gas transfer (Loyless and Malone, 1998; Hearn, 2009).

There are a variety of ways in which gas transfer is quantified in an airlift. Some of the ways to quantify how oxygen is transferred into the water are: the oxygen transfer rate (OTR), the standard oxygen transfer rate (SOTR), the actual oxygen transfer rate (AOTR), and the standard aeration efficiency (SAE). Some different ways to measure how much carbon dioxide is removed from water are the: carbon dioxide transfer rate (CTR), the standard carbon dioxide transfer rate (SCTR), the actual carbon dioxide transfer rate (ACTR), and the standard stripping efficiency (SSE).

The oxygen transfer rate and carbon dioxide transfer rate describe the mass of oxygen or carbon dioxide that is transferred into a volume of water over a period of time. This will vary depending on the diameter of airlift used, submergence and lift distances, as well as the temperature and salinity of the water. In an airlift, this is measured by taking the concentration of the dissolved gas going into and out of the airlift at a set water flow as shown below:

$$OTR = \left(C_{out}^{O} - C_{in}^{O}\right) * Q_L$$

$$CTR = \left(C_{out}^{C} - C_{in}^{C}\right) * Q_L$$

The SOTR and SCTR is a standardization of the length of time it takes an airlift to transfer gasses to their saturation concentration within a volume of water. The SOTR and SCTR are measured in clean, fresh water at a temperature of 20°C and barometric pressure of 1.00 atmosphere (ASCE, 1992). The equation for SOTR and SCTR are shown below:

$$SOTR = (K_L a_O)_{20} * [(C_S^O)_{20} - C_m^O] * V * 60 \frac{minutes}{hour} * 10^{-6} \frac{kg}{mg}$$

$$SCTR = (K_L a_C)_{20} * [(C_S^C)_{20} - C_m^C] * V * 60 \frac{minutes}{hour} * 10^{-6} \frac{kg}{mg}$$

More useful for the design of an airlift is the AOTR and ACTR as defined by Hearn (2009):

$$AOTR = K_O^{20.3} * \left(C_S^O - C_{in}^O\right) * Q_G$$
$$ACTR = K_C^{20.3} * \left(C_S^C - C_{in}^C\right) * Q_G$$

The AOTR and ACTR take the linear regression of an airlift's OTR/CTR vs. different gas flows to be able to calculate how much air is needed to deliver oxygen to a system. This allows the operator to know the amount of aeration in an airlift through the gas flow input. Once again, this only works for a specific lift, submergence, and pipe diameter and must be determined on site.

Finally, the SAE and SSE determines how efficient an airlift is at moving water. Once again, this value is tied to a specific lift, submergence, and pipe diameter combination. The power used by an airlift was defined in Loyless and Malone (1998) by:

$$kW_{ad} = 9.73 * 10^{-4} * Q_1 * P_1 * \left[\left(\frac{P_2}{P_1}\right)^{0.286} - 1\right]$$

The SAE and SSE are then:

$$SAE = \frac{SOTR}{kW_{ad}}$$

$$SSE = \frac{SCTR}{kW_{ad}}$$

Looking at the transfer rates of oxygen and carbon dioxide, Loyless and Malone (1998) Hearn (2009) and Alt (2010) found that gas transfer is oxygen limited; i.e. carbon dioxide stripping needs are surpassed by the oxygen transfer demand. This is partially to the presumption that deficits (the difference between the dissolved gases' water and saturation concentrations) are typically larger for carbon dioxide compared to oxygen. In normal seawater at 20°C oxygen saturation is 7.60 mg L^{-1} (Timmons et al., 2001) whereas CO_2 is 0.5 mg L^{-1}

(Hearn, 2009; Eshchar, 2003). In addition, the water quality recommendations give little room for error in aeration compared to degassing with typical aquaculture have the critical threshold concentration of dissolved oxygen at 5 mg L^{-1} compared to the critical threshold of CO_2 concentrations needing to reach 20 mg L^{-1} CO_2 for sensitive species (i.e. salmonids) (summarized in Colt et al., 2012; Alt, 2015).

2.6. DESIGN RULES FOR OPERATION

The location of the airlift will also be the determining factor in how effective it is at aeration and degassing, like any other gas transfer device. If the deficit is brought to its maximum value without interrupting the operation of the system, the air transfer can be greatly increased. Alt (2015) demonstrated that this increased deficit would make the airlift 250% more effective if it was located after the water flows through a biofilter and tank, as opposed to if it was between the tank and biofilter.

Additionally, Loyless and Malone (1998) concluded that airlifts should be sized for water flow rather than for aeration as the water flow requirements would be met long before the aeration requirements would be. However, it was found that the airlifts could aerate at 1/5 to 1/2 the rate of open water aeration systems. This added aeration rate allows for the airlift to give a substantial amount of supplemental aeration and degassing, lowering the cost of operating costs of the facility.

Taking the work of Loyless, Gudipati, Johnson, Hearn, and Alt; design rules for the airlift can be established for its operation and is shown in Table 2.1, which includes a safety factor of 50%. As stated by Loyless and Malone, the airlift should be designed for water circulation before focusing on air delivery. This design focuses on selecting a pipe that is large enough to maintain proper velocity for scour and solids control, which is 1.5 ft s^{-1} for the lift pipe and 2-3 ft

s^{-1} for the approach pipe, which is shown in Table 2.2. The water flow can be split between multiple pipes, which can greatly reduce the water velocity, and pipe sizes needed. The multiple airlifts can then be power from a common air blower or each given their own.

Table 2.1:Airlift design rules modified from Gudipati (2005)

Parameter	Criteria	Comment
Flow (Q)	GPM	Defined by system needs
Lift (L)	3- 15 inches	Defined by system needs
Lift pipe diameter	675 gpm ft^{-2} (1.5 ft s^{-1})	Diameter based upon net velocity; approach pipe based on balance of scour and headloss concerns
Approach pipe diameter	900-1350 gpm ft^{-2} (2-3 ft s^{-1})	
Injection depth (S)	>4*L in inches	Measured below the dynamic water level approaching the airlift
Air delivery (G)	0.275*Q_L	Set at a gas to liquid ratio of 1.3:1
Air delivery pressure	S+L in inches +10" (large systems)	Based upon depth of air injection + headloss in distribution

Table 2.2. PVC pipes selection for given water flows

Water Velocity (ft s^{-1})		1.5	2	3	1.5	2	3
		Water Flow (ft^3 s^{-1})			Water Flow (gal min^{-1})		
Pipe Size (SCH 40)	¼ inch	0.003	0.004	0.006	1.4	1.9	2.8
	¾ inch	0.006	0.007	0.011	2.5	3.3	5.0
	1 inch	0.009	0.012	0.018	4.0	5.4	8.1
	1 ¼ inch	0.016	0.021	0.031	7.0	9.3	14.0
	1 ½ inch	0.021	0.028	0.042	9.5	12.7	19.0
	2 inch	0.034	0.046	0.069	15.4	20.5	30.8
	2 ½ inch	0.050	0.066	0.100	22.4	29.8	44.8
	3 inch	0.077	0.103	0.154	34.6	46.1	69.1
	4 inch	0.133	0.177	0.265	59.5	79.4	119.0
	6 inch	0.301	0.401	0.602	135.1	180.1	270.1

Once the pipes size is selected, the submergence distance and total length of the draft tube is given by the total lift needed. This is a combination of the headloss from friction from

pipes and RAS operation as well as the static lift needed due to elevation changes. The final piece that needs to be selected is the air pump used. This is selected by looking at pump curves and selecting the best pump with a given air flow (1.3:1 Q_G:Q_L) and air pressure (S+ L+ reserve).

2.7. POSSIBLE ISSUES

While an airlift is usually reliable, it can encounter some issues due to design oversights including short circuiting and insufficient water flow. Short circuiting involves one airlift having a smaller submergence to overcome than other airlifts on the same air distribution line, usually due to inaccurate injection pipe placement. This lower pressure to overcome makes it a preferred path for air flow. The increased air flow lowers the density of the air/water mixture in an airlift draft tube, which moves more water, reducing the submergence, and, combined with the now higher backpressure of other airlifts, creates a feedback loop that can have all the air from the pump be diverted to one airlift. A solution for this is to either put in reduced fittings at each airlift air line to drive up the headloss due to friction or to put in a pressure relief valve in the main line which sets the pressure in the distribution line.

Insufficient water flow occurs when the water flow to the airlift is hampered or when air gets into the approach pipe. Normally, the water flows unimpeded to the air injection point where it pushes against the air/water mixture. If the water flow is reduced, the airlift is incapable of pulling water from the tank and its ability to move water to a higher elevation is greatly reduced. This can also occur if air gets into the approach pipe either from another process at the entrance of the approach pipe such as a waterfall or air backwash or from the airlift's air injection.

2.8. CONCLUSION

The airlift is a tool that allows an operator of an aquaculture facility to use one device to combine many design steps into one unit, reducing capital costs, saving energy, and reducing the risk of failure. The increased cost associated with continuous water circulation and extensive water treatment have prevented RAS from becoming more commercially accepted (Gutierez-Wing and Malone, 2006; Hearn, 2009). The airlift combines several processes used in aquaculture into one device: water movement, aeration, and degassing (Hearn, 2009). In a low head situation, not only does the airlift uses less energy per unit flow but also can use the same energy to regulate dissolved gasses. Consolidation can also be expanded to the pump itself, with one air blower servicing many tanks and keeping them in isolation. This has allowed the close loop RAS to compete with open flow through systems and continue to expand (Hird et al., 2000).

The improvements offered by an airlift are only possible if the facility design is properly designed and operated. Of high importance is the hydraulic design of a facility must minimize headloss to ensure efficient operation, including minimizing height changes and water velocities. Those operating the facility must be trained in how an airlift works and how to avoid issues in its operation. If these design concerns are met, then the airlift gives a low energy, simplified process train that can reduce capital costs.

The use of an airlift does not need to be limited to just aquaculture. The airlift excels in low lift situations where the movement of water can take advantage of the use of air. In the aquaculture industry, this is done by using the airlift to aid in the aeration and degassing of water which is needed to rear fish. In other applications, the use of air can be used to keep parts away from corrosive elements (namely salt water), but also centralize the motorized parts of a facility. This centralization does not compromise the disease isolation of the tanks in a facility as water is

not mixed between tanks. However, the airlift must be used in a way to take advantage of its unique design while limiting the areas it would be weak such as high lift or volatile compounds. Taking into consideration all of these pros and cons, the airlift can be a great tool for use in any fields and applications.

CHAPTER 3. DESIGN AND CONSTRUCTION OF MESOCOSMS

3.1. INTRODUCTION

The mesocosm experimental design is a balance between macrocosms and microcosms, both in size and design. Macrocosms are usually large-scale field studies in which one or two parameters are tracked across an area. Microcosms are usually small-scale experiments in which only a small subsection is analyzed. Mesocosms allow for more factors to interact than microcosms, while giving more control over a system than macrocosms. The ability to isolate variables is of great use to studies on the marsh given how many factors are intertangled (interactions of oil on the marsh shown as example in Figure 3.1.).

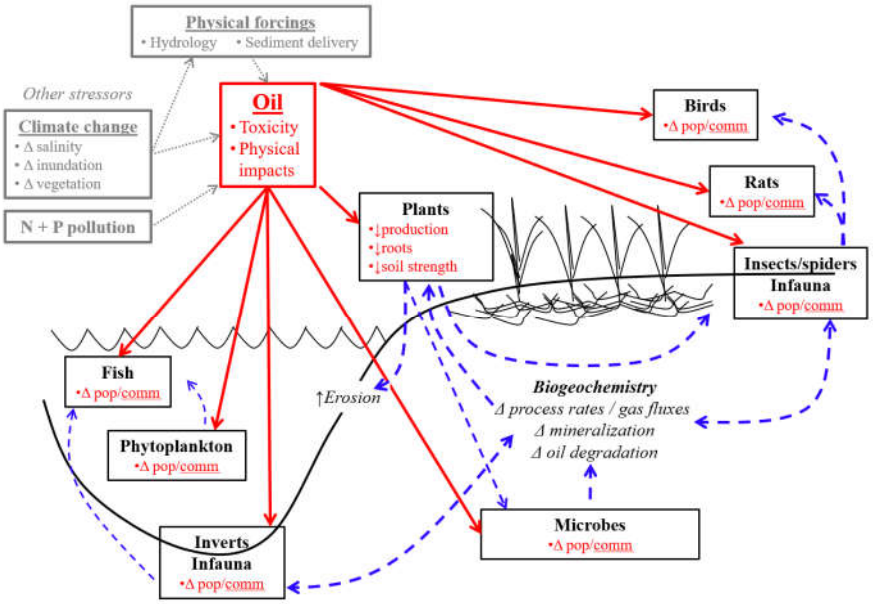

Figure 3.1. Direct and indirect impacts of oiling on marsh ecosystems (Rabalais et al., 2014). The direct impacts of oil and other stresses on several factors are shown in solid red lines with these factors' interactions shown in blue dotted lines.

Continuing to use oil spills as an example of how mesocosms can be used, the recalcitrant compounds found in oil are of concern when spills reach the marsh environment (Reddy and DeLaune, 2008; Mishra et al., 2012; Silliman et al., 2012; Hester et al., 2016). Studies on oil have run into issues due to the difficulty isolating impacts against variability of marsh sites and

conditions, and how heavily oiled sites are likely to have been lost after exposure (Rabalais et al., 2014). Macrocosms could be used to see how oil affects portions of the marsh but run into issues keeping track of individual parts of the spill. Microcosms could be of use to see how samples of oil breaks down under different conditions but run into issues scaling up. Mesocosms give the ability to have contained experiments that mimic actual conditions, eliminating the weaknesses of microcosms and macrocosms.

3.2. BACKGROUND

Microtidal environments are categorized by tides less than 2m with average flood velocities too low to import appreciable mineral sediment into their interiors (Kearney and Turner, 2016). These marshes are located worldwide and are typical of environments found in the Gulf of Mexico. Marshes located in these environments are vulnerable to stressors such as oil spills and climate change that impact the organic deposition of material.

Mesocosms have been used to study the marsh for several years with various control systems used. Both Elmgren and Frithsen (1982) and Dellagnezze et al. (2016) used mesocosms to study the effects of oil breaking down in a saltwater column with water pumps used to circulate the water. Lin and Mendelssohn (2012) and Judy et al. (2014) used marsh mesocosms to study oil's effect on plants in experiments that recycled water that drained through the soil and reapplied it to the marsh surface. Pennington et al. (2004) and Batubara (2014) used mesocosms to study the breakdown of chemicals in marsh soil with the tidal control. Pennington et al. (2004) accomplished using twelve water pumps while Batubara used four air pumps.

The airlift (Figure 3.2.) was first used in the mining industry in the 19th century and would later be used in aquaculture due to its ability to simultaneously circulate, aerate, and degas water (Castro et al., 1975; Loyless and Malone, 1998; Reinemann et al., 2001). The airlift's use

in aquaculture would lead to the development of design rules that could be used in other applications (Malone and Gudipati, 2005). These rules revolve around improving the energy efficiency of the airlift by limiting how far air could be injected (submergence, S) to move water a given height (lift, L). Once these two distances are set, the ratio of injected air flow (Q_L) to water flow (Q_G) moved can be found.

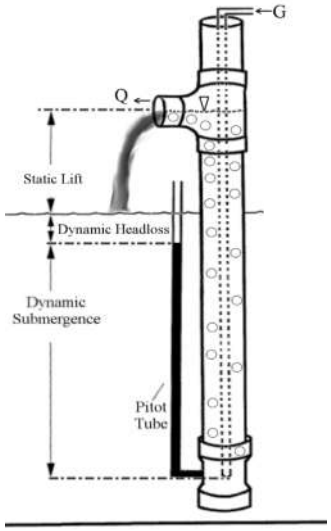

Figure 3.2. Airlift sections for an airlift used to move water to a higher elevation. When air is injected (G), the headloss from friction causes the submergence to decrease and the lift to increase

Castro and Zielinski (1980) were one of the first to use the S:L ratio alongside length and diameter of pipe to predict maximum liquid flow for a specific air injection. As air injection (Q_G) exponentially increases with a set S and L, the resulting fluid discharge (Q_L) will linearly increase until some optimal or peak air injection is met, at which water flow will slowly decline (Todoroki et al., 1973; Castro and Zielinski, 1980; Parker and Suttle, 1987; Wurts et al., 1994; Reinemann et al., 2001; and Hearn 2009). The water flow also increases with an increasing submergence depth given a constant lift and air flow, with most efficient S:L ratio found to be 3:1 to 4:1 (Wheaton, 1977), 4:1 (Timmons et al., 2001), and 4:1 to 5:1 (Malone and Gudipati, 2005).

To control the amount of water moved, a system of electronics based around the Processing© program, Arduino software and hardware, and the XBee software and hardware was developed to regulate the flow of air and water. The technologies used are relatively new with Processing© created in 2001, and the first Arduino board and XBee radio created in 2005 (Digi, 2018; Arduino, 2017; Processing©, 2018). Arduino Uno's are microprocessors that can be used to operate a wide array of electronics and sensors. The Arduino Uno and Processing© program are run from open source code (C++ and Java respectively), giving operators a great deal of control on a system's operation. XBee is a system of hardware and software run by Digi International that allow the integration of wireless connectivity.

3.3. MESOCOSM DESCRIPTION

Twelve mesocosms (Figure 3.3.) will be used by the CWC as a tool to continue to study the interaction of oil in a microtidal marsh with the mesocosms broken up into 4 groups: 3 oiling levels and a control in triplicate. The main study area consists of marsh plants and soil in nine-foot diameter sections deep enough to extend beyond the 1m (~3ft) rooting depth and are scheduled to experience a 15 cm (6 in.) amplitude tide and 10% exchange typical of coastal Louisiana (Turner, 2001). A support system brings water from the nearby bay to the mesocosms and treats experimental water before returning it to the marsh.

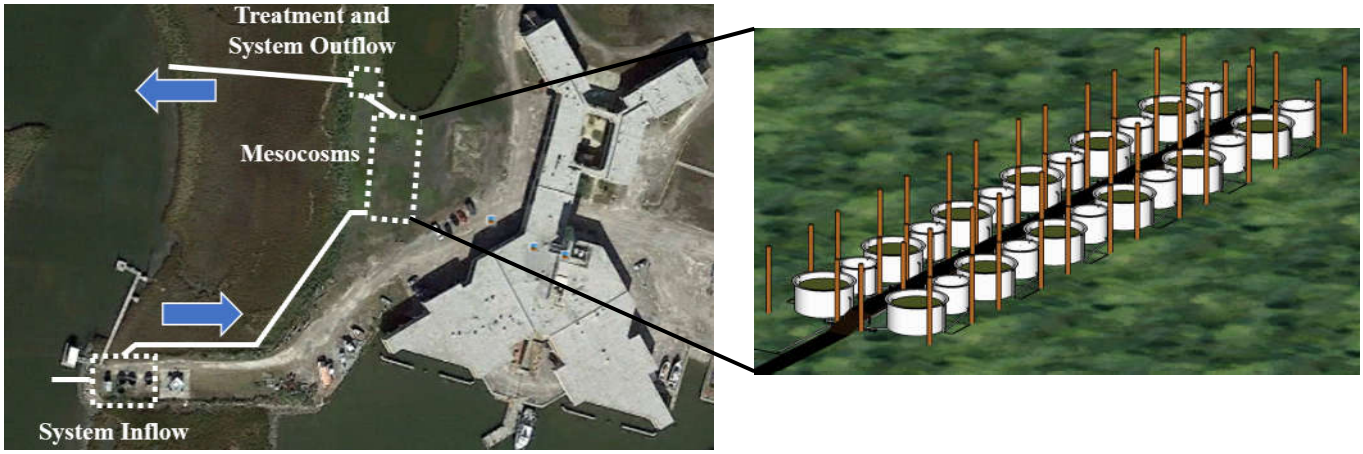

Figure 3.3. Mesocosm system at LUMCON.

The mesocosms (Figure 3.4.) were constructed in Cocodrie, Louisiana outside of the Louisiana Universities Marine Consortium (LUMCON). The marsh soil and plants are held in twelve 11.1 m^3 (~3,000 gal) 1.82m (6 ft) high, 3.04m (10 ft) diameter fiberglass tanks (referred to as the marsh tanks in this paper). Each marsh tank is connected by airlifts to a 4.0 m^3 (~1,000 gal) fiberglass surge tank 1.52m (5 ft) high with a diameter of 1.82 m (6 ft). The surge tank stores water for the tidal cycle and is covered with a shade cloth preventing excessive algae growth. The marsh and surge tanks have fiberglass ring 'anchors' on their base that prevent movement during flooding by cutting into the ground. Surrounding these tanks are wooden pilings and beams that support a boardwalk, bird netting, and mobile sampling walkways.

Figure 3.4. View of the two rows of mesocosms looking down from LUMCON Going from left to right, the tanks alternate between the marsh tanks and surge tanks

The marsh tanks (Figure 3.5.) were constructed to allow for a the marsh soil to be inundated up to a foot during high tide and for the marsh's perimeter to be exposed up to a foot during low tide. The marsh soil is placed atop a layer of sand and gravel, which provide drainage, along with a layer of permeable geocloth fabric between each layer. The marsh is held in the center of the tank by a fiberglass section in the shape of a cylinder that is ~1m (3 ft) high and 2.74m (9 ft) in diameter. The fiberglass insert forms a trough with the sidewalls of the fiberglass tank 0.30m (1 ft) deep, with the section touching the marsh soil perforated to allow for lateral water movement into the marsh. This trough contains 0.42 m^3 (13.9ft^3) of water while the water volume above the marsh surface is 1.85 m^3 (65.4ft^3). The bottom of the trough is set two

feet below the top of the marsh tank, allowing for a tidal amplitude of approximately 56cm (22 inches) with 5cm (2 inches) of freeboard.

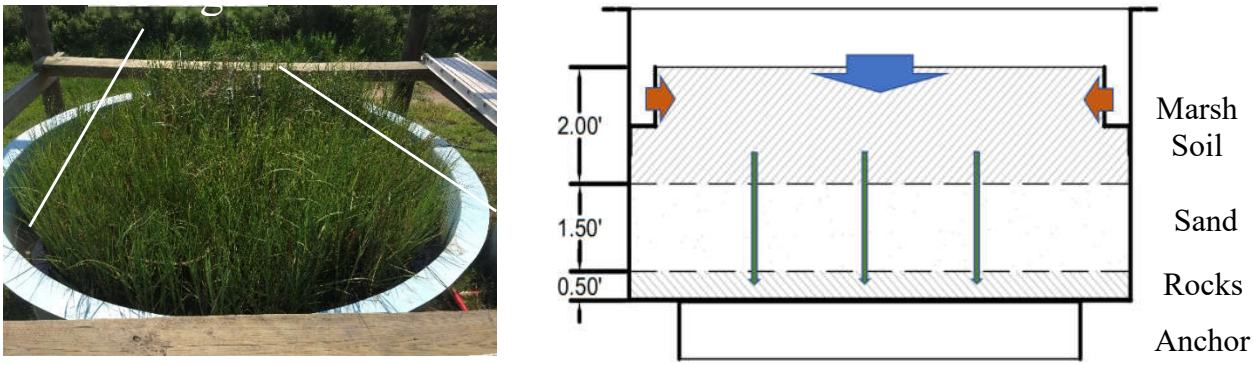

Figure 3.5. Marsh tank picture (right) and wire diagram (left). Water can either flow into the marsh horizontally during flooding (blue arrow), laterally through the sidewall of the trough (red arrows); or be drained through the sand and rocks (green arrows).

Tidal movement of water is based on a linearization of the diurnal tides in Terrebonne Bay (Figure 3.6.). The tidal water is exchanged between the marsh and surge tank by airlifts that are controlled by a process control system at two different rates to allow for a smooth transfer of water above and below the marsh surface.

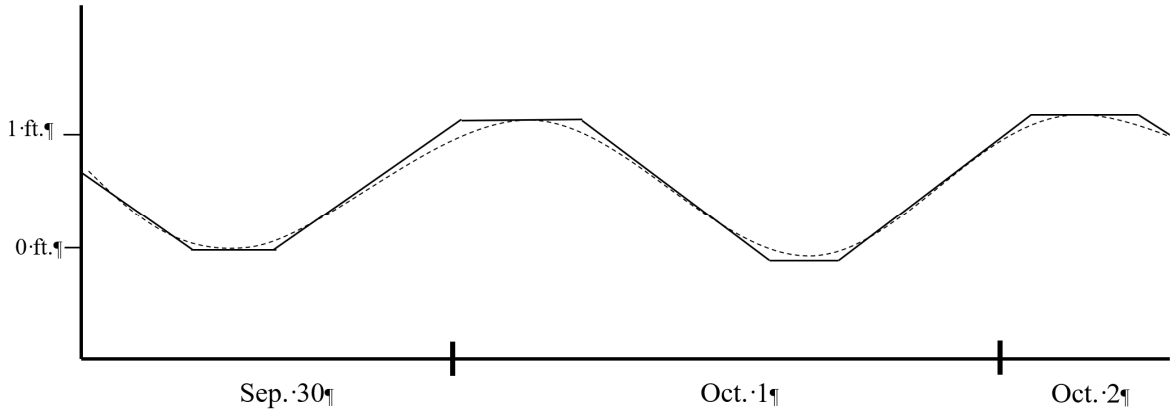

Figure 3.6. Linearization of tides in Terrebonne Bay showing the predicted tide (dotted line) and linear approximation (black line). Modified from Mobile Geographics, 2018)

3.4. WATER MOVEMENT

The movement of water in the mesocosms can be broken up into four stages of water movement: rising tide, falling tide, water removal, and water addition (Figure 3.7.). The rising and falling tides involve the exchange of water between the surge and marsh tank as the system dictates. This flow occurs at two different rates due to the two different cross-sectional areas above and below the trough surface. Water is exchanged with Terrebonne Bay water by first removing water from the surge tanks to the treatment system where it is discharged. After this, water must be added from the marsh to the surge tanks. All movements are controlled by motorized valves that regulate the flow of air or water. A breakdown of the pumping and power can be found in Appendix A.

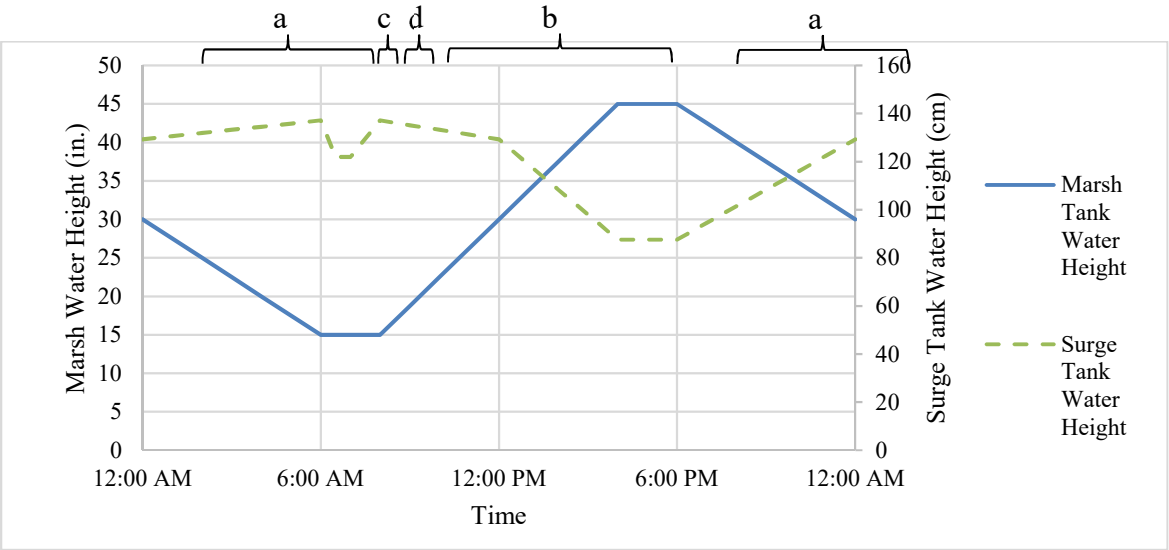

Figure 3.7. Water heights in marsh and surge tanks. Marsh tanks only have one cycle of water movement (tides, a & b) while the surge tanks have two (tides a & b, and water exchange c & d).

A variety of devices were used to move water throughout the system (Figure 3.8.). The main goal is to move new water into the mesocosms and move used water to the treatment system. Water is moved in and out of the system by two pipe networks spanning the three sections (bayou inflow, the mesocosm area, and the treatment overflow area). One leads from

the inflow area to the mesocosms and another goes from the mesocosms to the treatment outflow. This piping was designed to have an active pumping line and inactive anoxic line to control biofilm which is switched manually.

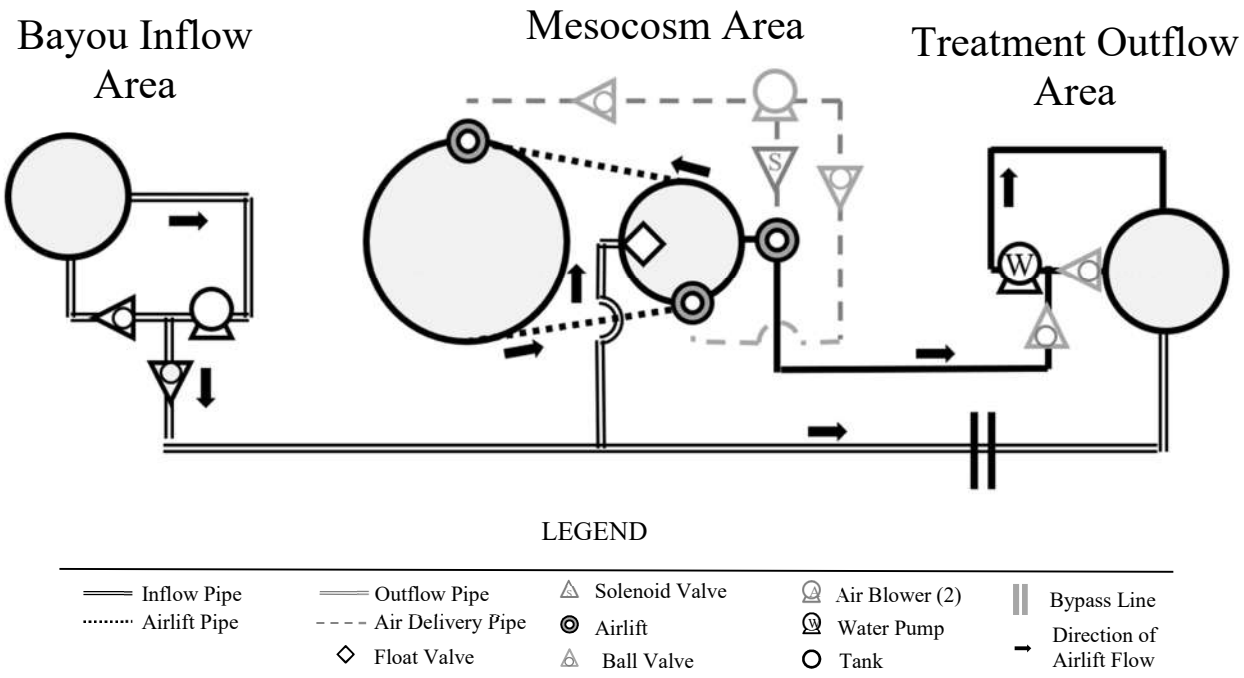

Figure 3.8. System components used in tidal movement and 10% water exchange. For clarity only one of the twelve mesocosms is shown

Water is pumped using different methods depending on the type of water movement needed (Table 3.1.). Water is moved by pumps across the long distances of pipe leading from the bayou to the mesocosms and from the mesocosms to the treatment system. Pumps were chosen due to their ability to move water across a great pressure difference and not corrode while in constant operation. Airlifts were chosen to move water at the mesocosms due to the low pressure needed to overcome and intermittent operation that could lead to a mechanical pump seizing while idle. All of these parts are controlled by motorized ball valves.

Table 3.1. System components used in water movements

Mesocosm Area / Water Movement	Mesocosm	Bayou Inflow	Treatment Outflow
Rising Tide	12 x Airlifts 2 x Air blowers 12 x Ball valves	-	-
Falling Tide	12 x Airlifts 2 x Air blowers 12 x Ball valves	-	-
10% Removal	12 x Airlifts 2 x Air blowers 12 x Ball valves	-	1 x Outflow pump 2 Ball valves
10% Addition	12 x Float valves	2 x Inflow pump 2 Ball valves	
Total Components	36 x Airlifts 36 x Ball valves 12 x Float valves	2 x Inflow pump 2 Ball valves	1 x Outflow pump 2 Ball valves

3.4.1. WATER EXCHANGE

The water in the mesocosms was designed to have a one-month turnover and replace water a rate of approximately 10% system volume per day, matching the hydraulic conditions of the bay. The water is brought to the mesocosms along the dual pipe line from the nearby channel of Terrebonne Bay and is removed by another dual pipe line that passes water through a treatment system before returning it back to the bay. The system works by directing the suction or discharge ends of the pumps to the mesocosms by motorized ball valves. To limit how much water comes into the system, float valves close off water flow to the surge tanks, redirecting the water to a bypass lines going to the treatment system. Stand pipes limit how much water can be taken out of the system in the drainage line.

The water is brought to the mesocosms by a single Sparus 160 SPET-4-AQ pump (Pentair 2018a) that can bring water to the system in under an hour with a 20% reduction in pipe diameter due to biofouling. The active pipe is designed to have a high-water velocity (3.8 ft s^{-1}) to remove any settled solids along the line. On account for the high-water velocity, ball valves

were used to control the direction of water flow to avoid water hammer. At the start of the 10% exchange during the marsh tank's low tide, water is removed from the surge tank and brought to the treatment detention tank (Figure 3.9). At the time of writing, this occurs passively using a standpipe whose opening is opened and closed by a motorized ball valve. The planned configuration will have water brought to a common drainage pipe from each surge tank by an airlift controlled by a solenoid. A pump loop at the outflow is connected to the drain line and aids in water removal to the treatment tank.

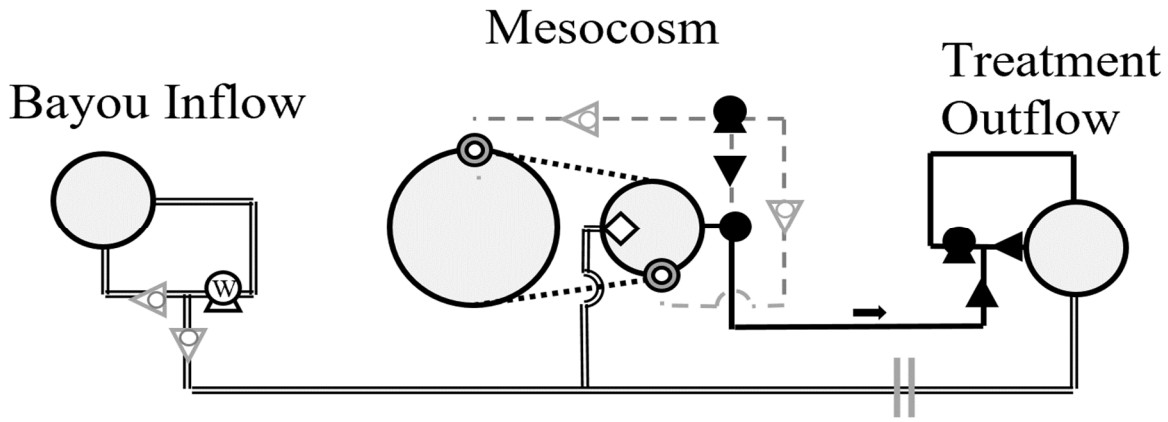

Figure 3.9. Water movement and components used for water removal. Components in use are in black: air blower, solenoid, airlift, ball valve, and treatment tank pump

At the end of the 10% exchange, water is added back into the system from the marsh inlet (Figure 3.10). The pump brings water to the mesocosms when the valves at the start loop switches from looping to the storage tank to bringing water to the mesocosms. This water fills the surge tanks through float valves. As these valves close, water can flow along a bypass line down to the treatment tank, which relieves water pressure as the surge tanks fill. Once the system detects that all water has been refilled, the ball valves at the inflow switch from bringing water to the mesocosms and back to the storage tank loop.

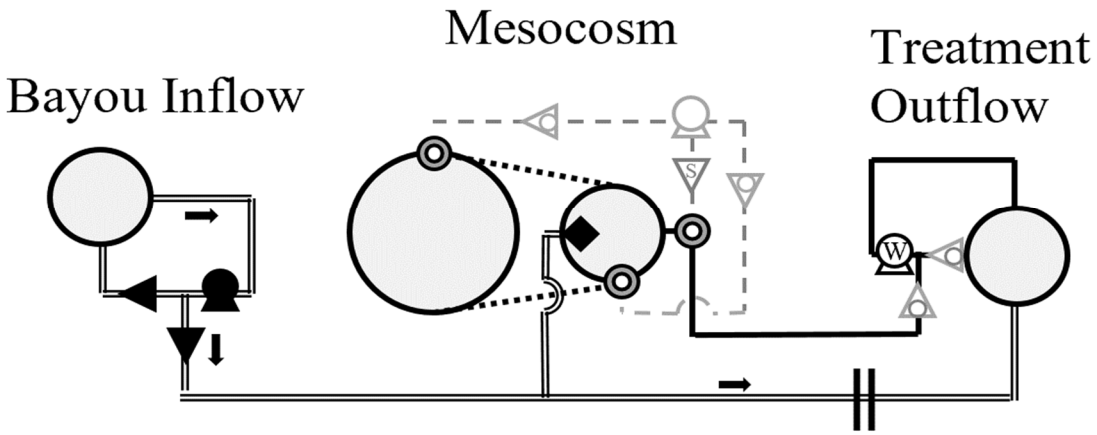

Figure 3.10. Water movement and components for 10% water addition. Components in use are in black: inlet pump, ball valve, float valve, and bypass line

3.4.2. TIDAL MOVEMENT

The tidal movement is accomplished at the mesocosms by two airlifts that move water between the marsh and surge tank by regulating the flow of air from two S41 Sweetwater® regenerative air blowers (Pentair 2018b). These blowers were designed to inject air into the base of the airlift (2.13m, 84 inches of pressure) at 9.4 L s^{-1} (20 ft^3 min^{-1}). Following design recommendations, a 1 ½ inch steel pipe connects the blowers to the main line to cool off the air coming from the blowers. The air then flows along a central distribution line where it is directed to airlifts by ball valves, with a 34.5 kPa (5 PSI) pressure relief valve venting excessive pressure along the line.

In tidal operations, only components at the mesocosms are engaged (Figure 3.11). During this process, the motorized valves supply air from the blowers to the marsh tank airlift, moving water from the surge tank to the marsh tank. A falling tide follows much of the same process, with the air blower supplying air to an airlift that is attached to the surge tank. The two airlifts were sunk to achieve a higher S:L ratio, with normal tides (±15cm) having a ratio from

1.6:1 to 4.7:1. At the highest and lowest possible tides, this ratio is 1:1 to 2.5:1. Current operation has each airlift operating at 1.6 GPM with the capability to increase to 5 GPM. This flow rate gives a velocity of 1.5 ft/s in a 1.5-inch SCH 40 PVC pipe, the minimum necessary for solids removal.

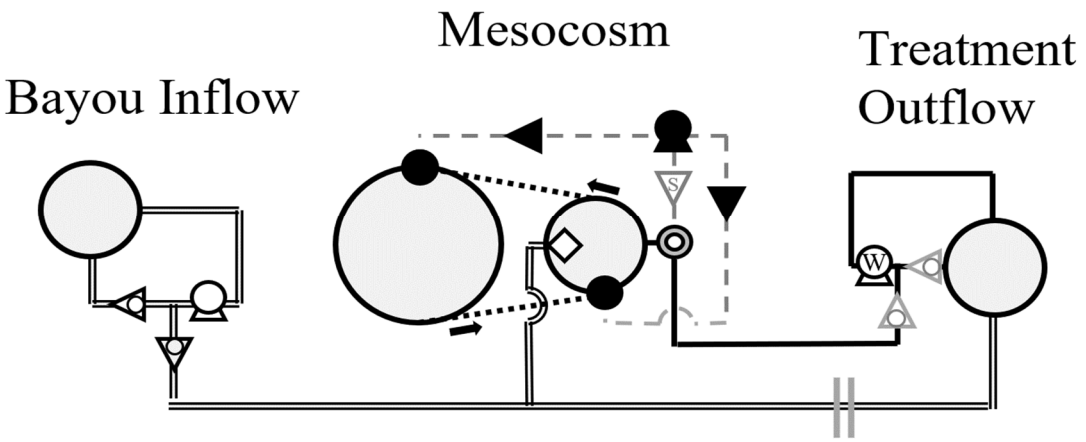

Figure 3.11. Water movement and components used when tide is raised. Components in use are in black: air blower, ball valve, and airlift

3.5.PROCESS CONTROL SYSTEM

The movement of water is controlled by a Processing© program that directs Arduino Uno's in the field to open or close the various valves. This program allows the user to schedule tides for years in advance or to operate the system manually. To run automatically, the system creates a tidal curve based on user input that is then used to decide what valves to open and close based on readings from water height sensors in the marsh and surge tanks. The process control system ignores this programmed tide when run manually and waits for user inputs to move water.

Communication between the Processing© program and Arduino Uno's is done wirelessly using the Digi XBee Pro Series 1 (code explained in Appendix B). This configuration is set up

in a star configuration with signals sent from a coordinator XBee connected to the computer running the Processing© program (Figure 3.12). Signals are sent to XBee's attached to the Arduino Uno's at each mesocosm and at the bayou inflow and treatment outflow. This configuration only allows the Arduino Uno's in the field to accept commands from the central computer, eliminating the possibility of a garbled communication from another Arduino Uno being received as command prompt.

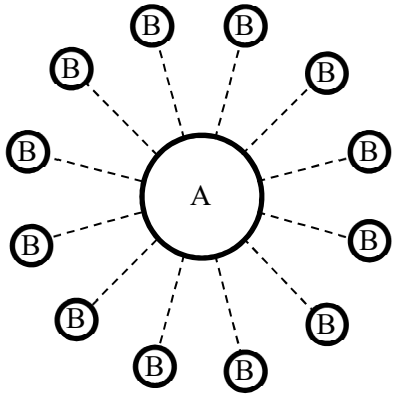

Figure 3.12. XBee star configuration communication set up with a single coordinator (A) and end devices (B).

The process control system also monitors and records the water height in each of the marsh and surge tanks of the mesocosms. The water level sensor in the marsh tank is a 2-ft chemical resistant Milone eTape and the surge tank has a MB7092 XL-MaxSonar-WRMA1 Maxbotix Weather-Resistant Ultrasonic Rangefinder (both pictured in Figure 3.13 with the logic found in Appendix C). These readings are taken every quarter-second with the system recording an average over 15 seconds. These readings are only sent to the main process control computer when queried by it and are recorded in a CSV file that is created for each day.

(a) (b)

Figure 3.13. Milone eTape (a) and Maxbotix Ultrasonic Range Finder (b)

The central computer program sends transmissions based on the tidal curve and time of day (logic breakdown in Figure 3.14 and code in Appendices D through G). Every 15 seconds, the program gathers information about the water height of each surge tank and marsh tank and records the data. After the user inputs what the tidal curve should look like, the tidal curve is broken up into a falling tide, rising tide, water removal, and water drainage. The process control program then checks the current time of day against this curve to decide what operation should be performed. The falling and rising tides check the marsh water height every minute to make sure that water is moving at the appropriate pace. The water removal and addition only occur once a day at low tide, and the water addition occurs after a short delay after the water removal ends.

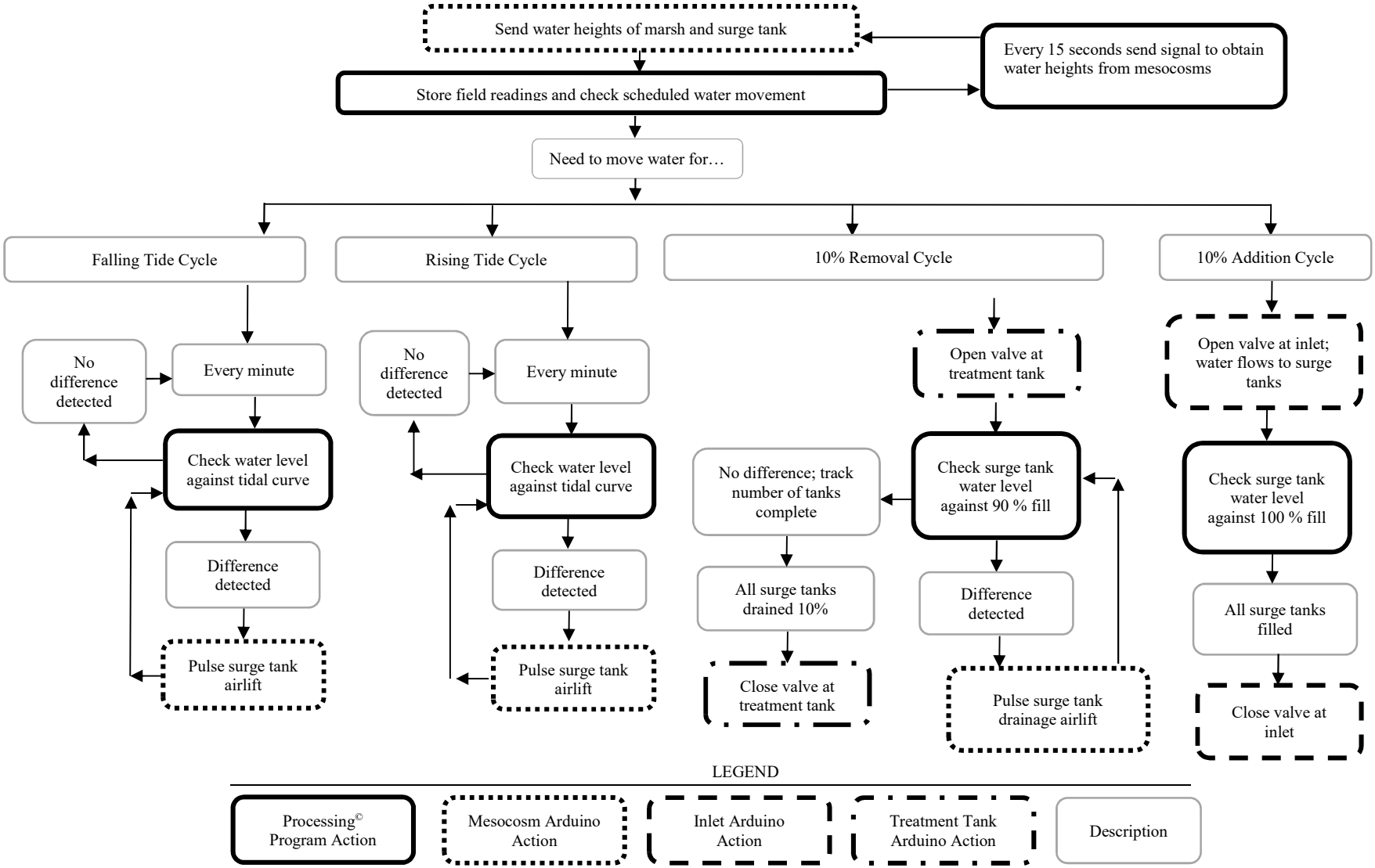

Figure 3.14. Flow chart for the Processing© program's logic.

3.6.RESULTS

For the mesocosms to mimic the marsh, the water levels must be able to match the programmed tide. The tides that was tested had a diurnal tidal pattern with a low tide of 15cm below the marsh surface (-15 cm) for 2 hours followed by a high tide of 15 cm above the marsh surface (+15cm) for 2 hours. These two tides occur six hours apart to give a 6-hour rising tide and a 14-hour falling tide to limit erosion of soil. This tidal variation was repeated to make sure the tides could repeat the same pattern.

The tidal variation over a single day is shown in Figure 3.15 in a single surge and marsh tank. In this example, the marsh tank water level was varied from a high to a low tide with water exchanged from the surge tank. The surge tank is not a mirror of this due to the 10% exchange. A standpipe is open from the start of the day until right before new water is delivered to the system. This standpipe is closed until the high tide of the marsh tank when the water level is below its opening. Water drains from this pipe once it is overtopped, which occurs at the end of the day in the figure.

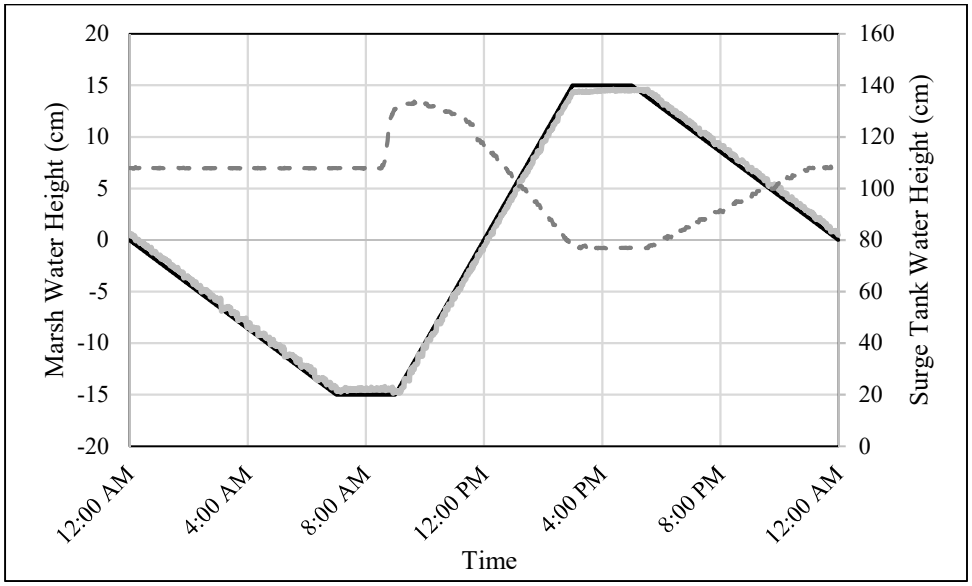

Figure 3.15. Tidal test in a single mesocosm tank over 24 hours showing readings of marsh tank (grey line) and the programmed tide (grey dashed line)

This tidal motion was repeated across all twelve marsh tanks (Figure 3.16). In this test, it can be seen how some sensors have more noise than others (mesocosms 1 and 11) and how some airlifts struggle due to air pressure differences (mesocosms 6 and 7). The noise is not expected to be an issue after sensor cleaning and calibration. The issues with air pressure should be fixed as the mesocosms transition from motorized ball valves to solenoid valves. Even with these slight issues, the test showed that the system showed that it could mimic a tidal pattern set by the operator, which is expected to become tighter as the system is worked on.

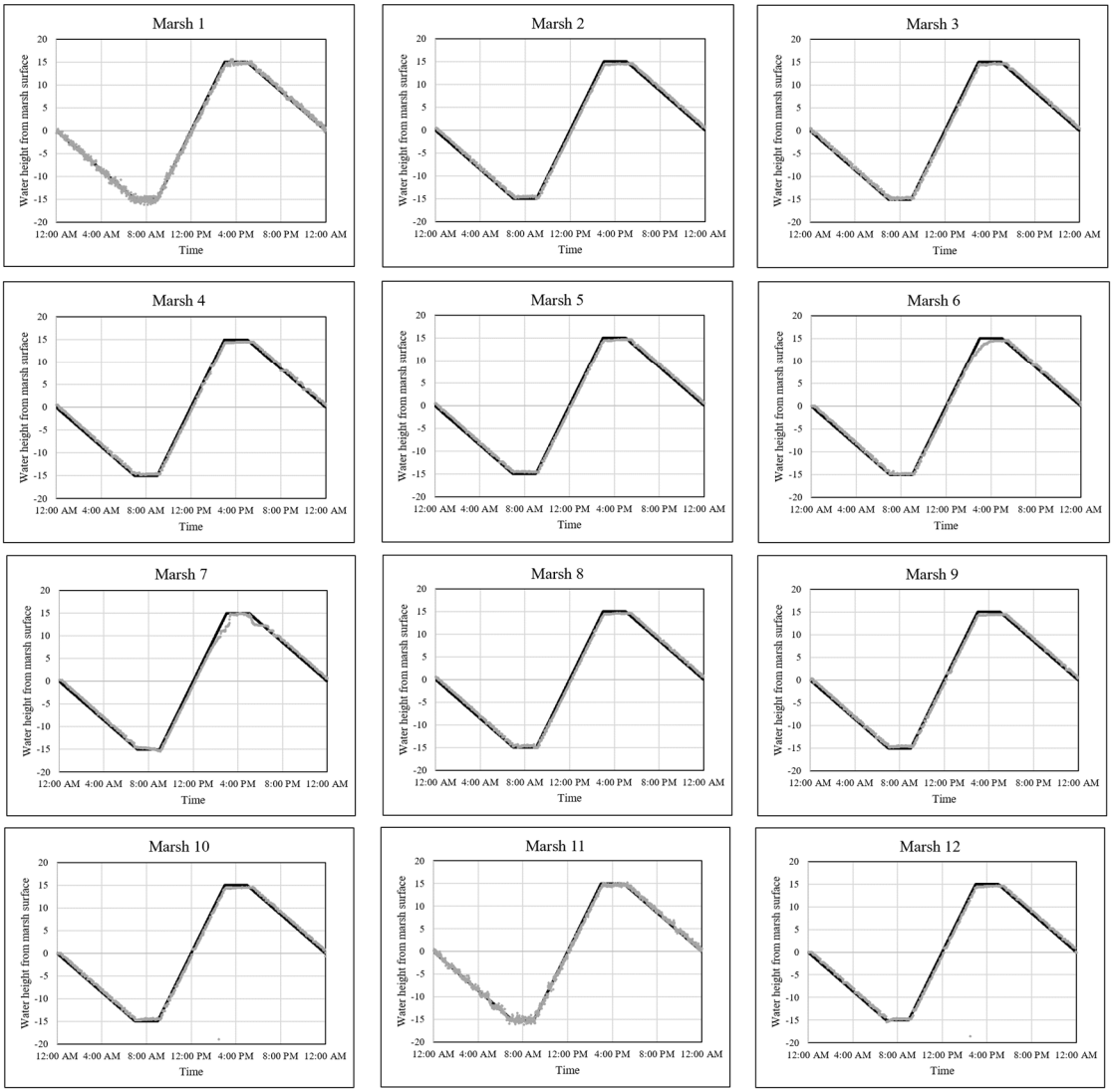

Figure 3.16. Rising and falling tide for all 12 mesocosms. Grey line shows the recorded readings and the black line is the programmed tide.

This test was repeated for 8 days to ensure that the process control system was able to match the calculated tide. Figure 3.17 shows the results for a single mesocosm's marsh tank. In this trial, water was not moved in the fifth day as the air piping was being worked on. This pattern is what would occur if experiments were carried out and the water needed to be still for a day. The pattern shown could also be changed to any number of high and low tides so long as the amplitude is less than 22cm and the fill or drain rate is less than 30 cm/hr.

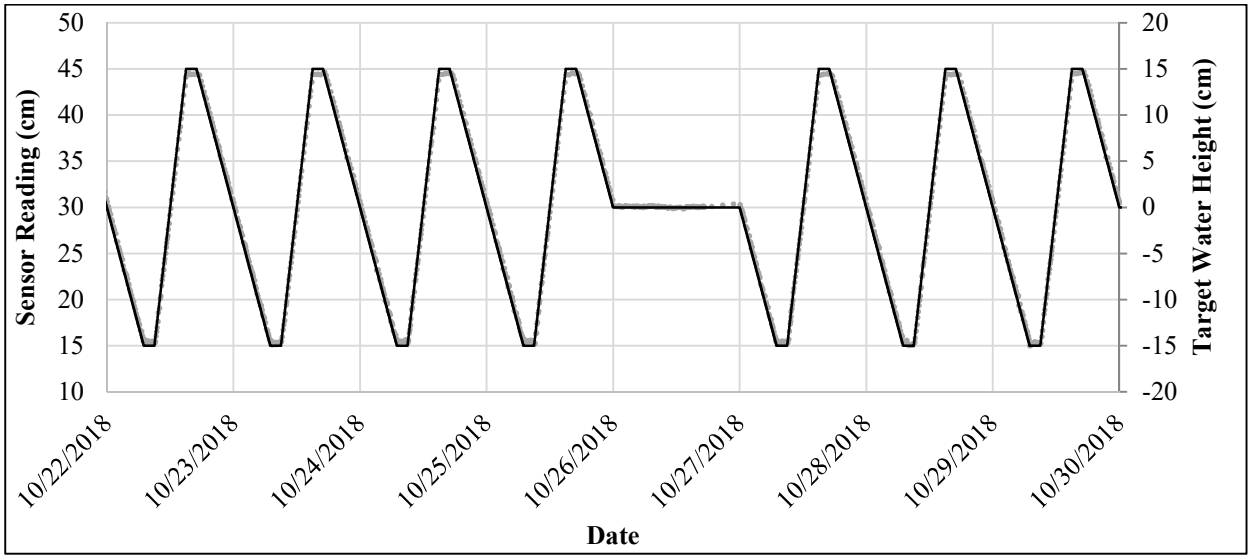

Figure 3.17. Recorded water height (grey dots) and target water height (black line) over 8 days in a marsh tank. On the 26th, no water was moved as the air delivery pipe was being worked on.

3.7.DISCUSSION AND CONCLUSION

The process control system allows for the construction of mesocosms that mimic the natural conditions of a microtidal marsh environment. The results from the initial run are encouraging and show that the system can control and replicate the system based on user inputs. The initial tidal tests create a trapezoidal tidal pattern and give a 10% exchange with the marsh. As the code is open source, the operators of the system have a great deal of control and ability to modify not only how the system runs but expand the data collection capacities.

While this system was created to mimic conditions found in Terrebonne Bay, Louisiana, the design can be modified to create a system for any microtidal environment for many different

types of studies. This includes not only areas with higher tidal amplitudes, but areas with semi-diurnal or mixed tides. The trapezoidal pattern used can also be switched to sinusoidal curves as the trapezoid was chosen for ease of user inputs. Additionally, advances in technology allow for electronic sensors to collect and store more data, supplementing field work.

The design of airlifts has several aspects that make it an ideal method to move water in this mesocosm design. With the mesocosm being open to the elements, reliability is needed, and the airlift gives the ability to have no mechanical parts in contact with water. It also acts as a check valve for tidal water moved from the surge tank to marsh tank, and for the air to be blown from a central location to distributed airlifts. This allows the twelve mesocosms to mimic the same tidal motion and exchange rate based on inputs from the operator of the system.

Care should be taken when using this design to develop other experiments. The airlift is a great tool but can change not only the composition of dissolved gasses, but volatile compounds in the water. With 40% weathered oil, this should not be of concern to this experiment, but could invalidate the results of others. The use of mechanical parts should also be weighed against the risk of breakdown or failure, with several steps taken to ensure their continual operation. The replacement and maintenance of parts of the system should also be included in the design to ensure that any breakdown can quickly be fixed.

CHAPTER 4. GLOBAL CONCLUSION

This dissertation describes the design and construction of a mesocosm that will be used to study a microtidal marsh. This design was made possible by airlifts that allow water to be moved in pulses with a centralized air blowing system that is kept away from salt water. The water movement of the mesocosm is controlled by the Processing© program with Arduino Uno microprocessors in the field.

While the airlift roots are in the mining and aquaculture industry, it has proved a robust tool in a different application. It's cost efficiency and ability to simultaneously degas, aerate, and circulate water have allowed it to find a use in aquaculture in low head applications. Its ability to operate several airlifts while only using a single air blower that can be kept out of the elements have made it an ideal tool to use in the design of the mesocosms. The ability to control the flow of water by regulating the flow of air also make it relatively easy to control by a process control system.

The mesocosms look to be a new tool for scientists to study the effects of oil on the marsh. One of the largest advantages is its ability to allow the operator of the system to monitor and control as well as adjust how the system operates. The control offered, not only extends to the amplitude and timing of the tides, but the amount of water exchanged with the marsh. A GUI allows the user of the process control system to create a tidal pattern, automating the ball valves that manipulate air and water flow. The commands from the central computer running the process control program are sent to Arduino Uno's operating various components of the mesocosm by a wireless network.

The mesocosm design is an indication of the evolution of electronics that has expanded the ability of scientists and engineers to design and conduct complex experiments. The

technology used in this project was only developed in the past few years and the amount of applications offered continues to expand. Being on the cutting edge does offer some drawbacks with experiments using new technology often being a field test in not only the literal sense, but also on the technology used. This need not be a deterrent so long as the weaknesses of the technology are known and accounted for. It is hoped that this dissertation lays out the ways in which the design of a microtidal mesocosm could be moved forward in both a new technology was used and how an older technology was used in a new application.

REFERENCES

3M. 2018. *Power Cable Splicing and Terminating Guide*. 3M, Accessed June 26, 2018. <https://multimedia.3m.com/mws/media/41103O/high-voltage-power-cable-splicing-terminating-brief.PDF>

Alt, D. 2015. Computer simulation of modern recirculation systems and their production strategies. A Master's Thesis. Louisiana State University. Baton Rouge, Louisiana. p. 76.

Alt, D., R. Tabor, and R. Malone. 2010. A Steady State Mass Balance Analysis of an Interim Airlifted PolyGeyser Recirculating System Design Criteria. Presentation at the Aquaculture Engineering Society. Roanoke, West Virginia, August 18-19, 2010.

Arduino, 2018. *Arduino Introduction.* Accessed July 26, 2018. <https://www.arduino.cc/en/Guide/Introduction>

Arduino. 2017. *About Us*. Arduino, accessed 26 November 2017, <https://www.arduino.cc/en/Main/AboutUs>

Banzi, M. 2010. *Dinner is Ready*. Arduino, Accessed 26 November 2017, <https://blog.arduino.cc/2010/09/24/dinner-is-ready/>

Barnea, D. and Y. Taitel. 1986. Flow Pattern Transition in Two-Phase Gas Liquid Flows. In: Encyclopedia of Fluid Mechanics, Gulf Publishing Co., Houston, TX, p. 404-469.

Batubara, D. 2014. Polycyclic aromatic hydrocarbon degradation in tidally influenced coastal marsh wetland studied in a laboratory mesocosm. PhD Dissertation. Louisiana State University. Baton Rouge, Louisiana.

Bellelo, S. Evaluation of Media Influence and Practical Applications for the Use of Static Low Density Media Filters in Domestic Wastewater Treatment. A Master's Thesis, Louisiana State University, Baton Rouge, Louisiana. p. 92.

Beyer, J., Trannum, H., Bakke, T., Hodson, P., and T. Collier. 2016. Environmental effects of the Deepwater Horizon oil spill: A Review. Marine Pollution Bulletin, 110(1), p. 28-51.

Castro W. E., and P. B. Zielinski. 1980. Pumping Characteristics of Small Airlift Pumps. In: Proceedings of the World Mariculture Society, 11. p. 163-174.

Castro, W., Zielinski, P., and P. Sandifer.1975. Performance characteristics of airlift pumps, World Mariculture Society Meeting, 6. p. 451-460.

Colt, J., Watten B., and T. Pfeiffer. 2012b. Carbon dioxide stripping in aquaculture – Part II: development of gas transfer models. Aquacultural Engineering 47, 38-46.

Costanza, R., d'Arge, R., De Groot, R., Farber, S., Grasso, M., Hannon, B., Limburg, K., Naeem, S., O'Neill R., Paruelo, J., Raskin, R., Sutton, P., and M. van den Belt. 1997. The value of the world's ecosystem services and natural capital. Nature, 387(6630), p. 253-260.

Dellagnezze, B., Vasconcellos, S., Angelim, A., Melo, V., Santisi, S., Cappello, S., and V. Oliveira. 2016. Bioaugmentation strategy employing a microbial consortium immobilized in chitosan beads for oil degradation in mesocosm scale. Marine Pollution Bulletin, 107(1), p. 107-117.

Digi International. 2018. *802.15.4 Release Notes*. Accessed Dec. 7, 2018. <http://ftp1.digi.com/support/firmware/93009382_F.txt>

Elmgren, R., and J. Frithsen. 1982. The use of experimental ecosystems for evaluating the environmental impact of pollutants: a comparison of an oil spill in the Baltic Sea and two long-term, low-level oil addition experiments in mesocosms. Marine Mesocosms. Springer US. p. 153-165.

Eshchar, M., Mozes N., and M. Fediuk. 2003. Carbon Dioxide Removal Rates by Aeration Devices in Marine Fish Tanks. The Israeli Journal of Aquaculture 55(2), 79-85.

Ferrarezi, R., Dove, S. and M. van Iersel. 2015. An automated system for monitoring soil moisture and controlling irrigation using low-cost open-source microcontrollers. HortTechnology, 25(1), p. 110-118.

Food and Agriculture Organization of the United Nations. 2014. The State of World Fisheries and Aquaculture. Food and Agriculture Organization of the United Nations. p. 243.

Gudipati, S. 2005. Distributed Airlift Systems Design with Application to Recirculating Soft Shell Crawfish Shedding Systems. A Master's Thesis. Louisiana State University, Baton Rouge, Louisiana. p. 84.

Gutierrez-Wing, M., and R. Malone. 2006. Biological filters in aquaculture: Trends and research directions for freshwater and marine applications. Aquacultural Engineering 34. p. 163–171.

Hearn, R. 2009. Gas Transfer in Air-Lifts Used to Recirculate Aquaculture Systems. A Master's Thesis, Louisiana State University, Baton Rouge, Louisiana. p.243.

Hester, M., Willis, J., Rouhani, S., Steinhoff, M., and M. Baker. 2016. Impacts of the Deepwater Horizon oil spill on the salt marsh vegetation of Louisiana. Environmental Pollution, 216, p. 361-370.

Hird, J., Sansalone, J., Beecher, L., and R. Malone. 2000. A design for siphoning U-Tube airlift pumps in closed loop recirculating aquaculture systems. Proceedings of the Water Environment Federation, WEFTEC 2000: Session 31 through Session 40, p. 572-581.

Huguenin, J., and J. Colt. 1992. Design and Operating Guide for Aquaculture Seawater Systems: First Edition. Developments in Aquaculture and Fisheries Science. Elsevier Science. Amsterdam, The Netherlands. p. 263.

Johnson, B. 2008. Airlift Assisted Wastewater Treatment. Master's Thesis, Louisiana State University, Baton Rouge, LA, p. 173.

Judy, C., Graham, S., Lin, Q., Hou, A., and I. Mendelssohn. 2014. Impacts of Macondo oil from Deepwater Horizon spill on the growth response of the common reed *Phragmites australis:* A mesocosm study. Marine pollution bulletin, 79(1), p. 69-76.

Kearney, M. and R. Turner. 2016. Microtidal marshes: Can these widespread and fragile marshes survive increasing climate–sea level variability and human action?. Journal of Coastal Research, 32(3), p.686-699.

Kearney, M., and R. Turner. 2016. Microtidal Marshes: Can These Widespread and Fragile Marshes Survive Increasing Climate–Sea Level Variability and Human Action?. Journal of Coastal Research: Volume 32, Issue 3: p. 686-699.

Lee, D., De Meo, O., Thomas, R., Tillett, A. and S. Neubauer. 2016. Design and construction of an automated irrigation system for simulating saltwater intrusion in a tidal freshwater wetland. Wetlands, 36(5), p. 889-898.

Lin, Q., and I. Mendelssohn. 2012. Impacts and recovery of the Deepwater Horizon oil spill on vegetation structure and function of coastal salt marshes in the northern Gulf of Mexico. Environmental Science & Technology, 46(7), p. 3737-3743.

Lockridge, G., Dzwonkowski, B., Nelson, R. and S. Powers. 2016. Development of a low-cost Arduino-based sonde for coastal applications. Sensors, 16(4), p. 528.

Loyless, C.J., 1995. A feasibility study of using airlift pumps for aeration, degasification, and water movement in recirculating aquaculture system. Doctoral dissertation, Thesis, Louisiana State University.

Loyless, J., and R. Malone. 1998. Evaluation of airlift pump capabilities for water delivery, aeration, and degasification for application to recirculating aquaculture systems. Aquacultural Engineering, Vol. 18, p. 117 –133.

Malone, R. 2017. Personal Communication.

Malone, R. and S. Gudipati. 2005. Airlift-PolyGeyser combination facilitates decentralized water treatment in recirculating marine hatchery systems. Aquaculture and Stock Enhancement of Finfish Proceedings of the Thirty-fourth U.S.-Japan Aquaculture Panel Symposium in Recirculating Marine Hatchery Systems. p. 43-50.

Malone, R., and S. Gudipati. 2007. Airlift-PolyGeyser® Combination Facilitates Decentralized Water Treatment in Recirculating Marine Hatchery Systems. Proceedings of the 34th US Japan Natural Resources Panel Aquaculture Symposium, San Diego, California. NOAA Technical Memorandum NMFS-F/SPO-85. October 2007.

Miller, L. and J. Long. 2015. A tide prediction and tide height control system for laboratory mesocosms. PeerJ, Vol. 3, p. 1442.

Mishra, D., Cho, H., Ghosh, S., Fox, A., Downs, C., Merani, P., Philemon,K., Jackson, N., and S. Mishra. 2012. Post-spill state of the marsh: Remote estimation of the ecological impact of the Gulf of Mexico oil spill on Louisiana Salt Marshes. Remote Sensing of Environment, 118, p. 176-185.

Mitsch W., and J. Gosselink. 2015. Wetlands. 5th ed. Hoboken, NJ; John Wiley & Sons, Inc.

Mitsch, W., Bernal, B., and M. Hernandez. 2015. Ecosystem services of wetlands. International Journal of Biodiversity Science, Ecosystem Services &Management, 11:1, p. 1-4.

Mobile Geographics. 2018. *Cocodrie, Terrebonne Bay, Louisiana Tide Chart*. Accessed October 1, 2018. <http://tides.mobilegeographics.com/locations/1304.html>

Nicklin, D.J., 1963, "The Air-Lift Pump: Theory and Optimization," Transactions of the Institution of Chemical Engineers, Vol. 41, p. 29-39.

Nixon, Z., Zengel, S., Baker, M., Steinhoff, M., Fricano, G., Rouhani, S., and J. Michel. 2016. Shoreline oiling from the Deepwater Horizon oil spill. Marine Pollution Bulletin, 107(1), p. 170-178.

Parker, N., and M. Suttle. 1987. Design of Airlift Pumps for Water Circulation and Aeration in Aquaculture. Aquacultural Engineering Vol. 6, p. 97-110.

Pennington, P., DeLorenzo, M., Lawton, J.C., Strozier, E., Fulton, M., G. Scott. 2004. Modular estuarine mesocosm validation: ecotoxicological assessment of direct effects with the model compound endosulfan. Journal of Experimental Marine Biology and Ecology 298, p. 369-387.

Pentair. 2018a. *Sparus 160 Energy-Efficient Aquaculture Duty 60 HZ Pumps*. Accessed July 26, 2018. < https://pentairaes.com/sparus-160-energy-efficient-aquaculture-duty-60hz-pumps.html>

Pentair. 2018b. *Sweetwater® Regenerative Blowers. Pentair Aquaculture*. Accessed July 26, 2018. <https://pentairaes.com/regenerative-blowers-aquaculture-duty.html>

Processing. 2018. *Overview. A short introduction to the Processing© software and projects from the community.* Accessed June 27, 2018. < https://processing.org/overview/ >

Rabalais, N., Woltman, S., Paruk, J., Bernhard, A., Parsons, M., Bracken-Grissom, H., Hooper-Bùi, L., Huang, H., Justić, D., Mariotti, G., Overton, E., Polito, M., Turner, R., Stouffer, P., Taylor, S., Roberts, B., Giblin, A., Able, K., Jensen, O., López-Duarte, P., Walker, A., Bensone, M., Shrestha, R., Kearney, M., Fodrie, J., and A. Engel. 2014. Proposal Prepared in Response to Gulf of Mexico Research Initiative Request for Proposals: Selection of Research Consortia. Coastal Waters Consortium. GoMRI2014-IV-958.

Ramsey, E., Meyer, B., Rangoonwala, A., Overton, E., Jones, C., and T. Bannister. 2014. Oil source-fingerprinting in support of polarimetric radar mapping of Macondo-252 oil in Gulf Coast marshes. Marine pollution bulletin, 89(1), p. 85-95.

Reddy, K., & R. DeLaune. 2008. Biogeochemistry of wetlands: science and applications. Boca Raton, Florida. CRC press. p. 774.

Reinemann, D. 1987. A Theoretical and Experimental Study of Airlift Pumping and Aeration with Reference to Aquacultural Applications. PhD Dissertation, Cornell University, Itacha, NY.

Reinemann, D., and M. Timmons. 1989. Prediction of Oxygen Transfer and Total Dissolved Gas Pressure in Airlift Pumping. Aquaculture Engineering 8, p. 29-46.

Reinemann, D.J., Hansen, J., Raabe, M., Mallison, J., and V. Byrd. 2001. Demonstration of airlift pump and lignocellulosics in recirculation aquaculture systems. Energy Center of Wisconsin, Madison. Wisconsin. p. 21.

Seeed Studio. 2018. *XBee Shield V2.0 Store Page*. Accessed July 19, 2018. <https://www.seeedstudio.com/XBee-Shield-V2.0-p-1375.html>

Silliman, B., van de Koppel, J., McCoy, M., Diller, J., Kasozi, G., Earl, K., Adams, P., and A. Zimmerman. 2012. Degradation and resilience in Louisiana salt marshes after the BP–Deepwater Horizon oil spill. Proceedings of the National Academy of Sciences, 109(28), p. 11234-11239.

Spotte, S. H., 1970. Fish and Invertebrate Culture. Wiley-Interscience Publication, John Wiley and Sons, New York, NY.

Stewart, R., Dossena, M., Bohan, D., Jeppesen, E., Kordas, R., Ledger, M., Meerhoff, M., Moss, B., Mulder, C., Shurin, J. and B. Suttle. 2013. Mesocosm experiments as a tool for ecological climate-change research. In Advances in ecological research.48. p. 71-181.

Timmons, M., and J. Ebeling. 2010. Recirculating Aquaculture 2nd Edition. Ithica, NY. Cayuga Aqua Ventures.

Timmons, M., Ebeling J, Wheaton F., Summerfelt S., and B Vinci. 2001. Recirculating Aquaculture Systems. Cayuga Aqua Ventures, Ithaca, NY. p. 650.

Todoroki, I., S. Yoshifusa, and T. Honda, (1973), Performance of Air-lift Pumps. Bulletin of ASME, 16. p. 733-741.

Turner, R. 2001. Of manatees, mangroves, and the Mississippi River: Is there an estuarine signature for the Gulf of Mexico?. Estuaries, 24(2), p.139-150.

Turner, R., Overton, E., Meyer, B., Miles, M., and L. Hooper-Bui. 2014. Changes in the concentration and relative abundance of alkanes and PAHs from the Deepwater Horizon oiling of coastal marshes. Marine pollution bulletin, 86 (1), p. 291-297.

Wheaton, F. 1977. Aquacultural Engineering, second printing. Robert E. Krieger Publishing Company. Malabar, Florida.

Wurts, W., McNeil., S., and D. Overhults. 1994. Performance and Design Characteristics of Airlift Pumps for Field Applications. World Aquaculture Society 24(4). p. 51-54.

APPENDIX A. PIPING AND POWER GUIDE

A process control system was set up to distribute water and air to where it is needed. This distribution system centralizes the water pumps and air blowers, cutting down on the number needed. The flow from the pumps and blowers is controlled by ball valves that move based on commands from the process control system. The distribution system is broken up into three lines: one for air and two for water. One water line leads from the nearby channel to the mesocosms and the other leads from the mesocosms to the treatment system. The piping for air brings air from the air blowers to the airlifts.

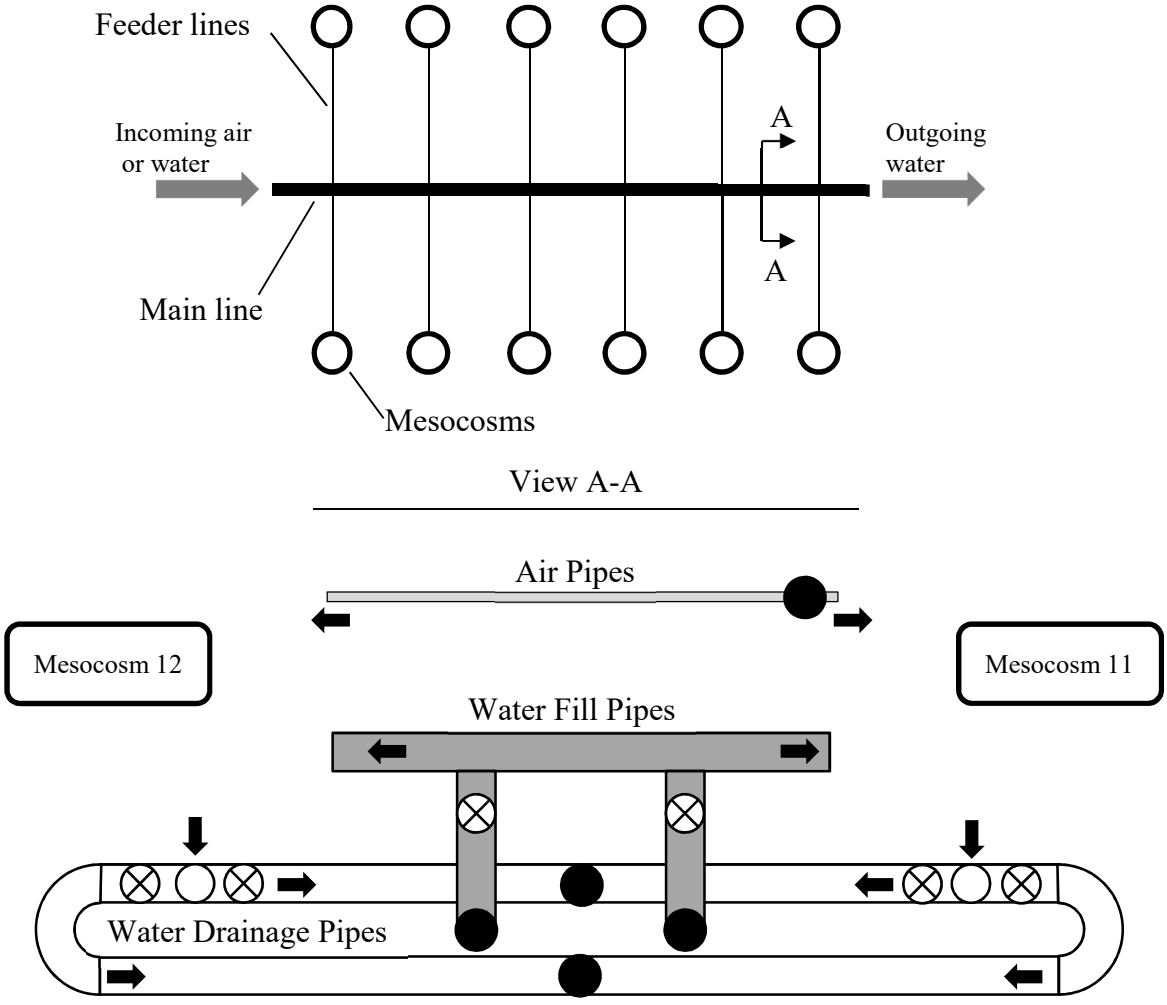

Figure A.1. Piping distribution system for air and water

The air distribution system (Figure A.2) directs air from the air blower to the airlifts located around the system. The air blowers are located at the same height as the walkway (four feet off the ground) to allow for easy maintenance and to be kept out of flood waters. The two S41 Sweetwater® regenerative air blowers are connected in series to be able to inject air at a depth of 94 inches (3.4 PSI) at 20 ft^3 per minute. This is enough pressure to overcome the initial water pressure of the airlift at enough flow to deliver 0.5 cm/min of lateral water movement in the mesocosms. Following design recommendations of the air blowers, a 1 ½ inch steel pipe connects the blowers to the main line to cool off the air from the blowers. The air then flows along a central distribution line where it is directed to airlifts by ball valves, with a 5 PSI pressure relief valve venting excessive pressure along the line. All 36 airlifts have a branch coming off this line with friction causing a pressure drop down the line. Normally, these airlifts would all have a restriction that would equalize the pressure seen by all airlifts, but the process control system acts as a regulator to the air flow.

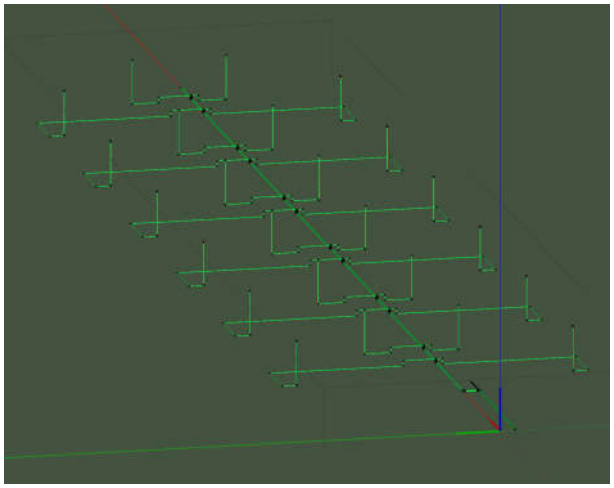

Figure A.2: Air distribution system

The mesocosm water exchange with the water from Terrebonne Bay uses two piping networks to move water: from the nearby Terrebonne Bay channel to the mesocosms and from the mesocosms to the treatment system (where it is eventually returned to the Bay). Water is

moved in enough volume to ensure that it has a turnover of one month which is about 10% per day. This exchange takes place in two stages: water draining from the mesocosms and water replaced from the marsh. Currently, water in the surge tank is removed down to a certain point by a stand pipe and is added back to a target height seen by a float valve. The height of the standpipe is 90% of the height of the float valve to give an average of 10% daily water exchange. Any water volume added from rain is also removed during the drain and fill cycle, keeping the mesocosm system water volume constant.

The water drainage system is composed of a central drain pipe that runs from each surge tank, under the boardwalk of the mesocosms and terminates at the treatment system. The pipe's inlets are standpipes that drain water to a set pipe in each surge tank. Water only drains from these pipes when a motorized valve on the central drain line is open. This line will eventually be changed to have 12 airlifts, one at each surge tank, giving the operator the ability to control how much water is drained. The main line will eventually have an in-line pump that helps to remove water quickly from the system.

The water entering the mesocosms comes from the nearby Terrebonne Bay channel and fills the surge tanks after the water drainage. This water is first pumped into a holding tank near the channel that removes large sediments through a strainer at the pump and from sedimentation. Water is pumped from this tank by a separate pump to the mesocosms with the process control system. Water flows into the surge tanks through float valves. As these valves are closed off, water pressurizes the pipe until it can flow to a bypass line that takes it to a treatment tank. Once all valves are closed, the flow to the mesocosms is shut off.

Power is supplied by a main line supplying power to the protected electronics at each mesocosm. A 120VAC line was run the length of the boardwalk (voltage drop of less than 3% with 10 AWG wire) and is converted to 9 VDC at each mesocosm by an adapter (Figure A.3). This DC voltage is then supplied to the electronics at each mesocosm (parts listed in Table A.1). All electronics except for the sensors are kept in junction boxes to shield them from rain and the elements. Wires are run to different junction boxes with non-metallic conduit secured by liquid tight push on connectors. All non-pin wiring connections were soldered together before being wrapped in electrical and splicing tape to prevent corrosion build up.

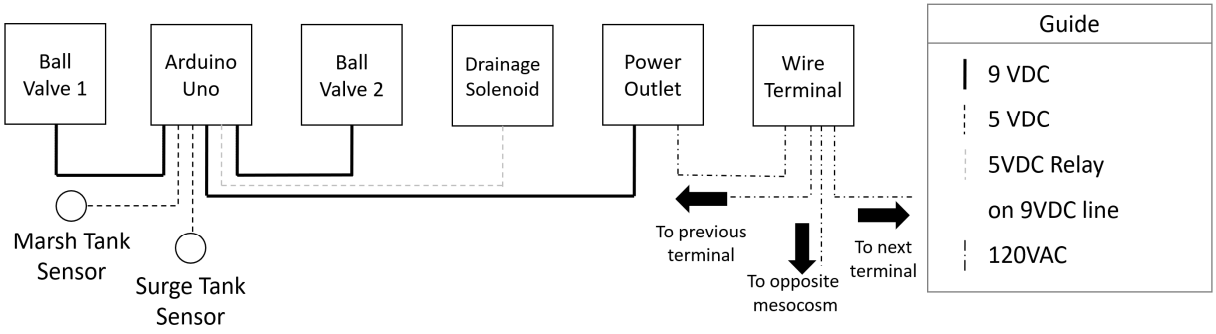

Figure A.3. Power diagram for various electrical components

Table A.1. Power requirements for various system components

Component	Voltage (VDC)	Amperage at supply voltage (mA)	Power Source
Arduino	5	1000	Power Outlet
Ball Valve (single)	9	220	Power Outlet
Solenoid	9	240	Power Outlet
XBee	3.3	250 (max)	Arduino
Ultrasonic Range Finder	5	20	Arduino
Milone eTape	5	20	Arduino

Steps were also taken to protect the mesocosms against severe weather. Protective measures were taken to shield the electronics from the elements as stated before. Electronics that

could flood (namely the outlet water pumps) were designed to be removed in case of hurricanes. The pilings surrounding the structure also block large debris from entering the mesocosm area and smashing into the pipe network or tanks of the mesocosm. If components are knocked out, spare parts are pre-assembled to allow for quick replacement.

APPENDIX B. WIRELESS COMMUNICATION

When the computer program needs to communicate to the 12 Arduino Uno's using a wireless network established by the Series 1 XBee's (API mode). The network running in API mode has several advantages including transmission confirmation, a checksum for packet integrity, and the ability of the XBee to control traffic to and from the Arduino Uno. This network consists of a central XBee that sends out signals to the XBee's out in the field which are connected directly to the Arduino Uno's which in turn control the various elements of the mesocosms. This network is setup in a star configuration as shown in Figure B.1. With the XBee connected to the main computer program acting as the coordinator and the Arduino Uno's in the field acting as end point devices.

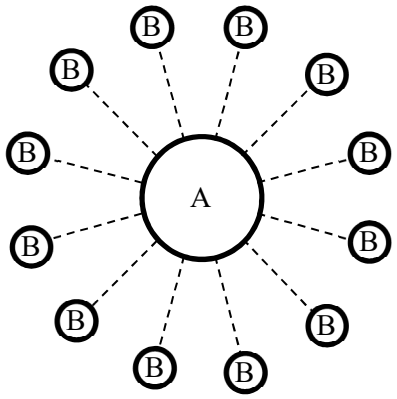

Figure B.1. XBee star configuration with a single coordinator (A) and 12 end devices (B).

This network is run in API mode to reduce the traffic that the end devices see, but the Processing© program cannot communicate with the Arduinos in the field directly due to the construction of data packets. This API transmission mode takes the data that would be sent and inserts it into packets that is structured to have a sending and receiving address as well as a sum check. Libraries are used to create data packets for the Arduino Uno that the attached XBee can

transmit. However, no supported library was found for the Processing© program to read or to send the API XBee transmissions. An intermediate Arduino Uno was used to get around this and allow communication between the computer and the Arduinos in the field as shown in Figure B.2.

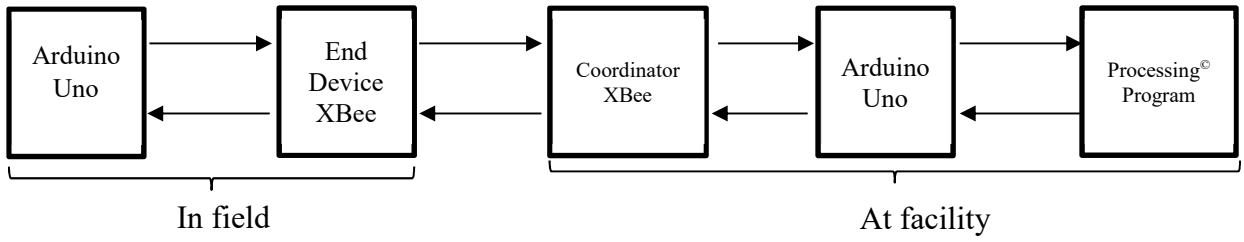

Figure B.2. Communication setup from Arduino Uno's to the Processing© program

APPENDIX C. SENSOR LOGIC

The only task that these microprocessors handle by themselves is the recording of the water height in the marsh and surge tank. The marsh tank has a 2-foot Milone eTape sensor to be able to record the full tidal motion, while the surge tank has an ultrasonic range finder to record the possible 5-foot change in water height. The ultrasonic range finder returns an analogue reading that is the distance in cm from the sensor located at the top of the tank to the water level. The Milone eTape only returns the voltage sent back from the sensor as it acts as voltage divider. The eTape is a resistive sensor whose resistance decreases as the water level increases. The water level (cm) in the marsh tank is given from the analogue reading by the formula:

$$h_{eTape} = \frac{R3 - \left(\frac{R1}{\left(\frac{A}{1023}\right)}\right)}{R2}$$

where:

h_{eTape}	is the water height reading from the eTape (cm)
$R1$	is a reference resistance based on the eTape (4000 Ω ± 20%)
$R2$	is a reference resistance based on the eTape (60 Ω ± 20%)
$R3$	is a reference resistance based on the eTape (8000 Ω ± 20%)
A	is the reading from the analogue pin, ranging from 0 to 1023 depending on the voltage range (0-5 V)

As these two sensors are exposed to the environment, some variability, or noise, in the readings are expected. This noise is usually counteracted by taking several readings and averaging them out to find out what the true readings should be. To reduce the amount of noise, the eTape was installed with a capacitor and both readings are taken every quarter second to create a 15 second average (Figure C.1 shows different methodologies used). This average is then transmitted to the Processing© program when called for.

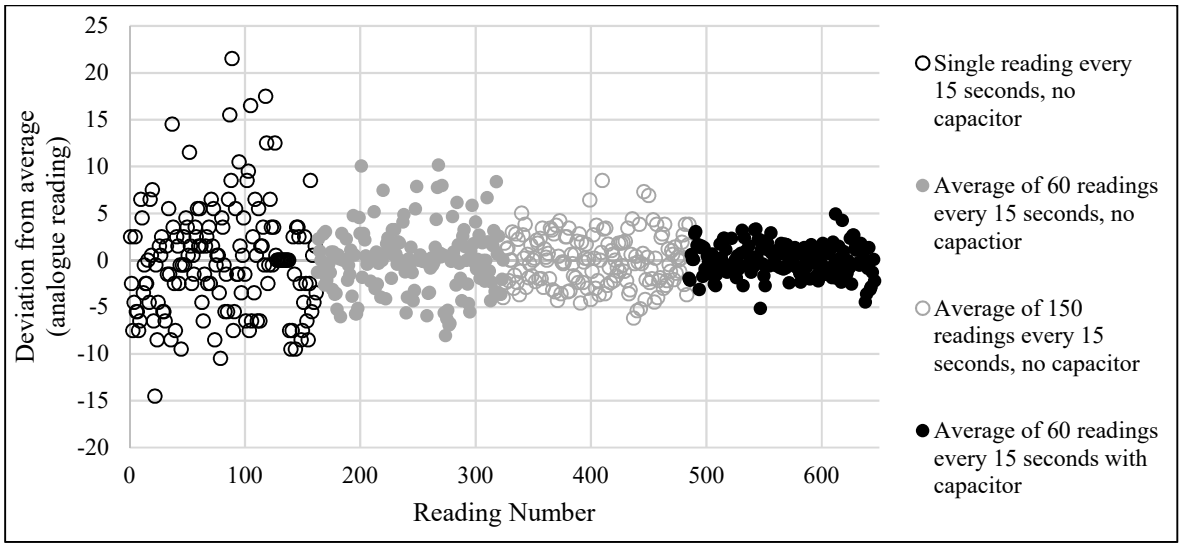

Figure C.1. Sensor variability with different set ups

APPENDIX D. "MIDDLE MAN" ARDUINO CODE

For the Processing© program to communicate with the Arduino Uno's out it the field, another Arduino Uno is used to convert the serial transmissions to an API framework. This framework is more complicated compared to transparently sending data (AT mode) but lets the XBee handle all routing procedures. The framework consists of three different parts: a series of frames that give information on what is being sent, the frame data being transmitted, and a final checksum frame. All data that is being sent is bracketed by the '<' and '>' characters which allow data to be easily parsed by the Arduino Uno's, leaving out all the other frame that are not needed.

Transmission to a specific Arduino Uno requires the XBee's 16-bit address to be known. Each XBee in the field has a unique ID ranging from 0x0000 to 0x0014, skipping the non-base 10 numbers. Numbers 0x0001 to 0x0012 are reserved for the communication to the mesocosm Arduino Uno's, 0x0013 is for communicating to the Arduino Uno controlling the outgoing water pump flow and 0x014 is for communicating to the Arduino Uno at the sedimentation tank pump. This transmission also has a built in receiving check, trying up to 3 times to resend the transmission packet if it is not received. If this fails, the XBee is configured to send a code specifying why.

```
#include <XBee.h>
#include <AltSoftSerial.h>

//SoftwareSerial SoftSerial(A0, A1);

XBeeWithCallbacks xbee;

AltSoftSerial SoftSerial;
#define DebugSerial Serial
//#define XBeeSerial SoftSerial
#define XBeeSerial SoftSerial

String A;
int comma1;
int comma2;
String action_n;
int action_n_int;
String meso_n;
int meso_n_int;
String Reading;
uint16_t Coor;
long count =0;
String trash2 ="";
int broadcast = 0;
float end_broadcast_check=0;
float broadcast_sum=0;
//define variable for sending info
int height_recording_trigger = 0;
int transmit_success=0;
int meso_reading_int = 1;
String sensor_readings [] = {"","","","","","","","","","","",""};
uint16_t Coor_Array[]={0xFFFF,0x0001,0x0002,0x0003,0x0004,0x0005,0x0006,0x0007,0x0008,0x0009,0x0010,0x0011,0x0012};
int sent_packet_int = 0;
int waiting_int=0;

void setup() {
//Start Serial Communication
Serial.begin(9600);
//Start XBee Communication
```

```
40    XBeeSerial.begin(9600);
41    xbee.begin(XBeeSerial);
42    delay(1);
43    }
44
45    void loop() {
46    /*
47    for(int i=0;i<12;i++){
48    if(sensor_readings[i].equals("")){
49    Serial.print("need");
50    Serial.print(i);
51    }else{
52    DebugSerial.print(sensor_readings[i]);
53    }
54    DebugSerial.print(",");
55    if(i==11){
56    DebugSerial.println(";");
57    }
58    }
59    */
60    label:
61    //Polling Loop for Tidal Height of Mesocosms.  Also sends
      information to computer.
62    if(height_recording_trigger==1 && broadcast == 0){
63    //exit condition in case of loop
64    count=count+1;
65    if(count>1000000){
66    for(int i=0;i<12;i++){
67    DebugSerial.print(sensor_readings[i]);
68    DebugSerial.print(",");
69    if(i==11){
70    DebugSerial.println(";");
71    }
72    }
73    height_recording_trigger=0;
74    count=0;
75    sent_packet_int = 0;
76    meso_reading_int = 1;
77    goto label;
78    }
79    if(sensor_readings[meso_reading_int-1].equals("")){
80    if(sent_packet_int == 0){
```

```
81    if(action_n_int==9){
82    A="<9>";
83    }
84    if(action_n_int==0){
85    A="<0>";
86    }
87    sent_packet_int=1;
88    transmit_success=0;
89    sendPacket(meso_reading_int);
90    }
91    if(transmit_success==1){
92    meso_reading_int=meso_reading_int+1;
93    sent_packet_int=0;
94    }
95    }else{
96    meso_reading_int=meso_reading_int+1;
97    }
98    //exit loop
99    //change meso reading int
100   if((meso_reading_int==3 && action_n_int==9) || (meso_reading_int==3
      && action_n_int==0)){
101   height_recording_trigger=0;
102   meso_reading_int=meso_reading_int-1;
103   for(int i=0;i<12;i++){
104   DebugSerial.print(sensor_readings[i]);
105   DebugSerial.print(",");
106   if(i==11){
107   DebugSerial.println(";");
108   meso_reading_int = 1;
109   waiting_int=0;
110   }
111   }
112   }
113   }
114   if(height_recording_trigger==1 && broadcast == 1){
115   broadcast=0;
116   height_recording_trigger=0;
117   end_broadcast_check=0;
118   broadcast_sum=0;
119   waiting_int=1;
120   sendPacket(0);
121   }
```

```
122  // Checks to see if the computer (Serial) sends something
123  if(Serial.available()>0){
124  String trash = String(Serial.readStringUntil('<'));
125  String txt = String(Serial.readStringUntil('>'));
126  //Read Parts of String
127  comma1 = txt.indexOf(',');  //finds location of first comma
128  meso_n = txt.substring(0, comma1);   //captures first mesocosm
     String
129  meso_n_int=meso_n.toInt();
130  comma2 = txt.indexOf(',', comma1+1 );   //finds location of second
     comma,
131  action_n = txt.substring(comma1+1,txt.length()-1);   //captures
     second action String
132  action_n_int=action_n.toInt();
133  if(meso_n_int==0){
134  broadcast=1;
135  }else{
136  broadcast=0;
137  }
138  A="<"+String(action_n_int)+">";
139  //checks to see if there is a call for tidal heights.  If not, tell
     the Arduinos to open or close a valve.
140  height_recording_trigger=0;
141  if(action_n_int==9 || action_n_int==0){
142  for(int i=0;i<12;i++){
143  sensor_readings [i] = "";
144  }
145  height_recording_trigger=1;
146  count = 0;
147  }
148  if(height_recording_trigger==0){
149  sendPacket(meso_n_int);
150  }
151  }
152  // Checks to see if the Arduinos (XBeeSerial) have sent anything
153  if(XBeeSerial.available()>0){
154  String trash = String(XBeeSerial.readStringUntil('<'));
155  String txt = String(XBeeSerial.readStringUntil('>'));
```

```
156   //Read Parts of String
157   comma1 = txt.indexOf(',');  //finds location of first comma
158   meso_n = txt.substring(0, comma1);   //captures first mesocosm
      String
159   comma2 = txt.indexOf(',', comma1+1 );   //finds location of second
      comma,
160   Reading = txt.substring(comma1+1,txt.length()-1);   //captures
      second mesocosm String
161   if(isValidNumber(meso_n)==true && action_n_int==0){
162   if(meso_n.toInt()>40){
163   int place_holder_int = meso_n.toInt()-41;
164   sensor_readings[place_holder_int]=Reading;
165   transmit_success=1;
166   }else{
167   int place_holder_int = meso_n.toInt()-31;
168   sensor_readings[place_holder_int]=Reading;
169   transmit_success=1;
170   }
171   }else if(isValidNumber(meso_n)==true && action_n_int==9){
172   if(meso_n.toInt()>30){
173   int place_holder_int = meso_n.toInt()-31;
174   sensor_readings[place_holder_int]=Reading;
175   transmit_success=1;
176   }else{
177   int place_holder_int = meso_n.toInt()-21;
178   sensor_readings[place_holder_int]=Reading;
179   transmit_success=1;
180   }
181   }
182   for(int i=0;i<12;i++){
183   broadcast_sum=sensor_readings[i].toFloat()+broadcast_sum;
184   }
185   if(broadcast_sum==end_broadcast_check){
186   height_recording_trigger=1;
```

```
187   }else{
188   end_broadcast_check=broadcast_sum;
189   broadcast_sum=0;
190   }
191   }
192
193   }
194   //call for packet to be sent to mesocosms over the XBee network
195   void sendPacket(int meso_ID) {
196   //Send TX16 Request for Series 1 XBee
197   Tx16Request txRequest;
198   Coor = Coor_Array[meso_ID];
199   txRequest.setAddress16(Coor);
200   char test[int(A.length()+1)];
201   A.toCharArray(test,int(A.length()+1));
202   uint8_t payload[sizeof(test)];
203   for(int i=0;i<sizeof(test);i++){
204   payload[i]=test[i];
205   }
206   txRequest.setPayload(payload, sizeof(payload));
207   // And send it
208   uint8_t status = xbee.sendAndWait(txRequest, 5000);
209   if (status == 0 && height_recording_trigger==0 && waiting_int==0) {
210   DebugSerial.print("S");
211   } else if(height_recording_trigger==1 && status == 0) {
212   } else if(height_recording_trigger==0 && waiting_int==0) {
213   DebugSerial.print("F");
214   DebugSerial.print(status, HEX);
215   DebugSerial.print("!");
216   }else{
217   sent_packet_int = 0;
218   }
219   }
220   //check to make sure numbers are a string so the program does not
      crash
221   boolean isValidNumber(String strg){
222   for(byte i=0;i<strg.length();i++)
223   {
224   if(isDigit(strg.charAt(i))) return true;
```

```
225     }
226     return false;
227     }
```

APPENDIX E. ARDUINO HOOKUP GUIDE

While Arduino Uno's are used in every area of the mesocosm, the connections of each differ. All the microprocessors require connection to the XBee's but differ in the number of sensors and motors that they are required to control or monitor. The Arduino Uno's in the field need to control 3 valves and monitor 2 sensors. The ones that control the flow of water into and out of the system only control 2 valves. The final Arduino Uno is used as a middle man, translating the commands from the Processing© program to transmittable XBee packets, and vice versa.

The Arduino Uno's in the field require connections to the various sensors and motorized valves they control as well as connections to the XBee RX and TX lines. This set up uses or reserves 8 of the digital pins and all 6 of the analogue pins of the Arduino Uno as shown in Figure E.1. Digital pins 0 and 1 are reserved for communication to the Arduino Uno to upload code. Six of the remaining digital pins are used by the wireless DC motor driver shield (D3 to D8) to control the motors. The first two analogue pins (A0 and A1) are used for communication with the XBee with the last two (A4 and A5) used for I2C communication to the QWIIC boards. The final two analogue pins (A2 and A3) are used for communication with the ultrasonic range finder and Milone eTape sensors.

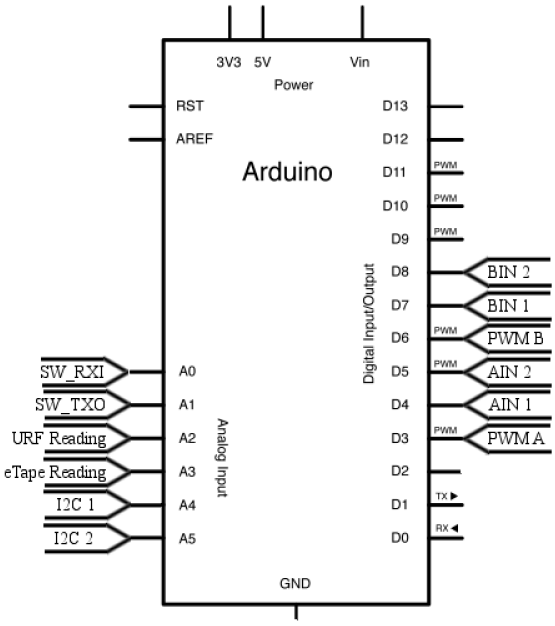

Figure E.1: Arduino Uno pin connections

The wireless motor driver shields used by the Arduino Uno's allow for the microprocessor to control the system without having to wire an integrated circuit. The only connections used in this shield are the 2 sensor connections, the 4 wires for positive and ground connections for the DC motor, and the I2C connection used by the QWIIC connection. The motors are powered by an external power source connected to the shield's barrel jack. The analogue sensor connections are powered from the Arduino Uno's 5V connection. The shield's XBee toggle is set to SW_SER to avoid bricking the XBee, and the power toggles are set to VS and IOREF modes to route the 9V external power source to the DC motors and not the sensors.

The Arduino Uno's that control the water flowing into and out of the system do not have any sensors attached and only control the 2 DC motor valves that direct the water flow (Figure E.2). The only wires that need to be connected are those for the motor are plugged into the shield's A and B motor connections. The remaining connection to the XBee RX and TX pins are handled by the board with the operator only ensuring that the toggles on the shield are set to the

correct position (SW_SER, VS, and IOREF). A diagram showing the pin connections is shown in

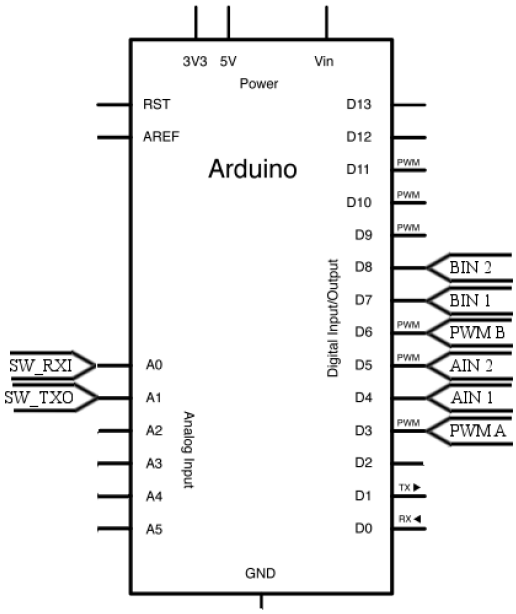

Figure E.2. Inflow and outflow Arduino Uno connections

The coordinator Arduino is only expected to transmit and receive data along the computer serial port and the XBee. As the Arduino used to communicate between the computer and Arduinos in the field does not operate a motor, a XBee shield by i Studio is used (Seeed Studio 2018). This frees up the digital pins and expands the number of libraries that can be used for serial communication, making the altsoltserial library ideal as it allows for simultaneous transmission and receiving, but requires digital pins 8 and 9 on the Arduino Uno (Figure E.3). These pins are used to control the motor with the wireless DC motor shield. The only wiring or routing that must be done with the Seeed shield is that the RX and TX pins must be set to 8 and 9. This Arduino Uno is not run from a separate power source and is instead connected to the computer with a USB-B adapter.

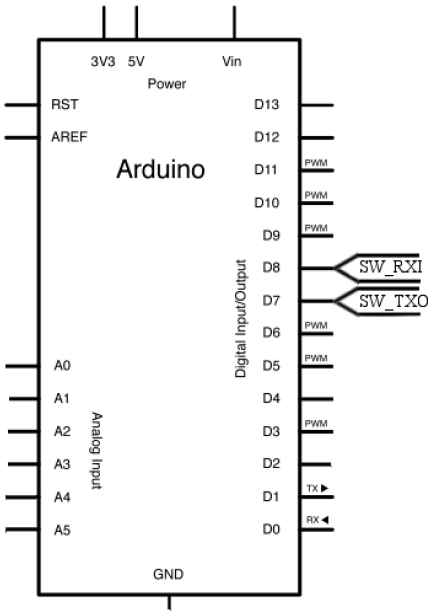

Figure E.3. Middle man Arduino Uno connections

APPENDIX F. FIELD ARDUINO CODE

The code used by the Arduino Uno's in the field constantly records water heights from the marsh or surge tank while waiting for signals from the main computer program before operating valves or sending back readings. The signals coming in from the main program are first parsed with the '<' and '>' symbols used as markers. If the water heights are requested by the main computer, the readings are sent back along with the number of the mesocosm, with the two values separated by comma's and bracketed once again by the '<' and '>' symbols.

To prevent the airlifts from moving too much water, the valves controlling the flow of air are only opened for a small amount of time. This is handled by the Arduino Uno automatically closing a valve after a set amount of time has passed after it has opened. Code blocking is avoided by not using the delay() function and instead recording the height when a valve is opened. A check using the if() function closes the valve once enough time has passed.

```
#include <XBee.h>
#include <AltSoftSerial.h>
XBeeWithCallbacks xbee;
AltSoftSerial SoftSerial;
#define DebugSerial Serial
#define XBeeSerial SoftSerial
String A;
int comma1;
int comma2;
String action_n;
int action_n_int;
String meso_n;
int meso_n_int;
String Reading;
uint16_t Coor;
long count =0;
String trash2 ="";
const byte numChars = 32;
char receivedChars[numChars];
char tempChars[numChars];
int broadcast = 0;
float end_broadcast_check=0;
float broadcast_sum=0;
//define variable for sending info
int height_recording_trigger = 0;
int transmit_success=0;
int meso_reading_int = 1;
String sensor_readings [] = {"","","","","","","","","","",""};
uint16_t Coor_Array[]={0xFFFF,0x0001,0x0002,0x0003,0x0004,0x0005,0x0006,0x0007,0x0008,0x0009,0x0010,0x0011,0x0012};
int sent_packet_int = 0;
int waiting_int=0;
unsigned long broadcast_millis;
int send_trigger=0;
int broadcast_reset_trigger = 0;
char messageFromPC[numChars] = {0};
int integerFromXBee = 0;
float floatFromXBee = 0.0;
boolean newData = false;
int meso_number_of_calls = 0;
void setup() {
```

```
41          //Start Serial Communication
42          Serial.begin(9600);
43          //Start XBee Communication
44          XBeeSerial.begin(9600);
45          xbee.begin(XBeeSerial);
46          delay(1);
47          }
48          void loop() {
49          if((millis()-broadcast_millis>2000 && send_trigger==1 &&
        height_recording_trigger==1)){
50          for(int i=0;i<12;i++){
51          DebugSerial.print(sensor_readings[i]);
52          DebugSerial.print(",");
53          if(i==11){
54          DebugSerial.println(";");
55          }
56          }
57          send_trigger=0;
58          height_recording_trigger=0;
59          }
60          /*
61          for(int i=0;i<12;i++){
62          if(sensor_readings[i].equals("")){
63                  sendPacket(int(i+1));
64          }
65          meso_reading_int=meso_reading_int+1;
66          }
67          */
68          label:
69          if((millis()-broadcast_millis>500 &&
        height_recording_trigger==1 && broadcast_reset_trigger==1)){
70          meso_reading_int=1;
71          broadcast_reset_trigger=0;
72          }
73          //Polling Loop for Tidal Height of Mesocosms.  Also sends
        information to computer.
74          if(height_recording_trigger==1 && millis()-
        broadcast_millis>750){
75          //exit condition in case of loop
76          count=count+1;
77          if(count>1000000){
```

```
78          for(int i=0;i<12;i++){
79          DebugSerial.print(sensor_readings[i]);
80          DebugSerial.print(",");
81          if(i==11){
82          DebugSerial.println(";");
83          }
84          }
85          height_recording_trigger=0;
86          count=0;
87          sent_packet_int = 0;
88          meso_reading_int = 1;
89          goto label;
90          }
91          //^^loop
92          if(meso_reading_int<14){
93          if(sensor_readings[meso_reading_int-1].equals("")){
94          //Serial.print("need");
95          //Serial.println(meso_reading_int);
96          if(sent_packet_int == 0){
97          //Serial.print("Alpha");
98          sent_packet_int=1;
99          transmit_success=0;
100         sendPacket(meso_reading_int);
101         }
102         if(meso_number_of_calls>2){
103         meso_reading_int=meso_reading_int+1;
104         count=0;
105         meso_number_of_calls=0;
106         transmit_success=0;
107         }
108         if(transmit_success==1){ //if there is a successful
    transmission
109         meso_number_of_calls=0;
110         sent_packet_int=0;
111         }
112         }else{
113         meso_reading_int=meso_reading_int+1;
114         }
115         }
116         }
117         if(height_recording_trigger==1 && broadcast == 1){
118         broadcast=0;
```

```
119         meso_reading_int=1;
120         end_broadcast_check=0;
121         broadcast_sum=0;
122         waiting_int=1;
123         broadcast_reset_trigger=1;
124         sendPacket(0);
125         }
126         // Checks to see if the computer (Serial) sends something
127         if(Serial.available()>0){
128         String trash = String(Serial.readStringUntil('<'));
129         String txt = String(Serial.readStringUntil('>'));
130         broadcast_millis=millis();
131         waiting_int=0;
132         send_trigger=1;
133         //Read Parts of String
134         comma1 = txt.indexOf(',');  //finds location of first comma
135         meso_n = txt.substring(0, comma1);   //captures first mesocosm
    String
136         meso_n_int=meso_n.toInt();
137         comma2 = txt.indexOf(',', comma1+1 );   //finds location of
    second comma,
138
            action_n = txt.substring(comma1+1,txt.length()-1);   //captures
    second action String
139         action_n_int=action_n.toInt();
140         if(action_n_int==0 || action_n_int==9){
141         broadcast=1;
142
143         }else{
144         broadcast=0;
145         }
146         A="<"+String(action_n_int)+">";
147         //checks to see if there is a call for tidal heights.  If not,
    tell the Arduinos to open or close a valve.
148         height_recording_trigger=0;
149         if(action_n_int==9 || action_n_int==0){
150         for(int i=0;i<12;i++){
```

```
151         sensor_readings [i] = "";
152         }
153         height_recording_trigger=1;
154         count = 0;
155         }
156         if(height_recording_trigger==0){
157         sendPacket(meso_n_int);
158         }
159         }
160         // Checks to see if the Arduinos (XBeeSerial) have sent
    anything
161
            //taken from http://forum.arduino.cc/index.php?topic=396450
162         recvWithStartEndMarkers();
163         if (newData == true) {
164         strcpy(tempChars, receivedChars);
165         // this temporary copy is necessary to protect the original
    data
166         //   because strtok() used in parseData() replaces the commas
    with \0
167         parseData();
168         newData = false;
169         }
170         }
171         //wait until data is received
172
            //taken from http://forum.arduino.cc/index.php?topic=396450
173         void recvWithStartEndMarkers() {
174         static boolean recvInProgress = false;
175         static byte ndx = 0;
176         char startMarker = '<';
177         char endMarker = '>';
178         char rc;
179         while (XBeeSerial.available() > 0 && newData == false) {
180         rc = XBeeSerial.read();
181         if (recvInProgress == true) {
182         if (rc != endMarker) {
183         receivedChars[ndx] = rc;
184         ndx++;
185         if (ndx >= numChars) {
186         ndx = numChars - 1;
```

```
187         }
188         }
189         else {
190         receivedChars[ndx] = '\0'; // terminate the string
191         recvInProgress = false;
192         ndx = 0;
193         newData = true;
194         }
195         }
196         else if (rc == startMarker) {
197         recvInProgress = true;
198         }
199         }
200         }
201         //parse data
202         //taken from http://forum.arduino.cc/index.php?topic=396450
203         void parseData() {      // split the data into its parts
204         char * strtokIndx; // this is used by strtok() as an index
205             strtokIndx = strtok(tempChars,",");      // get the first part
    - the string
206             integerFromXBee = atoi(strtokIndx);     // convert this part to
    an integer
207             strtokIndx = strtok(NULL, ",");
208             floatFromXBee = atof(strtokIndx);     // convert this part to a
    float
209             if(integerFromXBee>40){
210             int place_holder_int = integerFromXBee-41;
211             sensor_readings[place_holder_int]=floatFromXBee;
212             meso_reading_int=meso_reading_int+1;
213             transmit_success=1;
214             }else{
215             int place_holder_int = integerFromXBee-21;
216             sensor_readings[place_holder_int]=floatFromXBee;
217             meso_reading_int=meso_reading_int+1;
218             transmit_success=1;          //trigger to let Arduino know there
    was a transmission
```

```
219         }
220         }
221         //call for packet to be sent to mesocosms over the XBee network
222         void sendPacket(int meso_ID) {
223         //Send TX16 Request for Series 1 XBee
224         Tx16Request txRequest;
225         Coor = Coor_Array[meso_ID];
226         txRequest.setAddress16(Coor);
227         char test[int(A.length()+1)];
228         A.toCharArray(test,int(A.length()+1));
229         uint8_t payload[sizeof(test)];
230         for(int i=0;i<sizeof(test);i++){
231         payload[i]=test[i];
232         }
233         txRequest.setPayload(payload, sizeof(payload));
234         // And send it
235         uint8_t status = xbee.sendAndWait(txRequest, 5000);
236         if (status == 0 && height_recording_trigger==0 &&
    waiting_int==0) {
237         DebugSerial.print("S");
238         } else if(height_recording_trigger==1 && status == 0) {
239         } else if(height_recording_trigger==0 && waiting_int==0) {
240         DebugSerial.print("F");
241         DebugSerial.print(status, HEX);
242         DebugSerial.print("!");
243         }else{
244         sent_packet_int = 0;
245         meso_number_of_calls=meso_number_of_calls+1;
246         }
247         }
248         //check to make sure numbers are a string so the program does
    not crash
249         boolean isValidNumber(String strg){
250         for(byte i=0;i<strg.length();i++)
251         {
```

```
252             if(isDigit(strg.charAt(i))) return true;
253         }
254         return false;
255         }
```

APPENDIX G. PROCESSING© CODE

The user of the program can manually control the water level of the mesocosms or have the program automate it. The automated program requires the user to input data to build the tidal pattern. The tidal curve is broken into two tidal sections to create a tidal pattern and needs six inputs: two amplitudes, two durations, and two amplitude times (Figure G.1), with the amplitude measured from the marsh surface. Calculations are performed on these inputs to generate the required water height for every minute of the day. These values are then checked against actual water heights to determine if a valve needs to be opened.

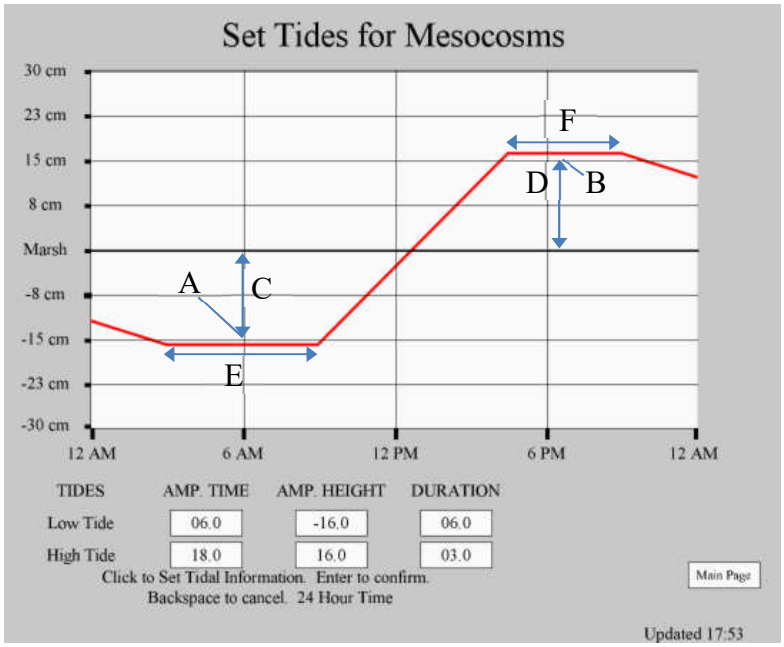

Figure G.1. GUI for operator to input variables for tides. The six inputs are the two amplitude times (A and B), two amplitudes (C and D), and two durations (E and F)

As discussed in the main document, the main point of the Processing© computer program is to create a GUI that the operator of the system can then use to set a tidal pattern for the mesocosms. Behind this are several functions that create files for data to be stored, called up, and manipulated. The data that is stored is mainly the water heights of the surge and marsh tanks

over time along with the calculated water heights and tidal height readings that were recorded. Processing© allows for the program files to be split into functions that can be called up by the main program. Currently, there are nine functions:

1. mesocosm_water_height
   This is the main portion of the program with the setup() and draw() functions contained within. This is also where most of the variables are created.
2. tidal_motion
   This function determines what kind of tide is being created (rising, falling, or no tide) and sends a signal to the Arduino Uno's in the field to operate a valve if needed
3. record_heights
   This function takes the water readings from the tanks and stores them
4. meso_tide_set_page
   This function creates the page where the mesocosm tides are set
5. Manual_Override
   This function creates a page where the mesocosms can be set to automatic or manual, as well as having the user turn on or off valves manually
6. main_page
   This is the function that creates the main page where the user can navigate to all other pages as well as see the water levels of the mesocosm
7. individual_meso_page
   This function creates the page that shows the water level of a mesocosm over the course of the day as well as what the calculated tidal pattern is
8. file_check
   This function creates files for data to be saved (valve positions, tidal amplitudes, and mesocosm data)
9. file_check_named
   This function uses a lot of the same code as file_check, but only looks for a named function in the file check

mesocosm_water_height

```
1   //import all libraries
2   import processing.serial.*;
3
4   //Name Serial Port and associated variables with reading it
5   Serial xbee;
6   String myString = null;
7   String myStrings = "";
8
9   //variables used to move water
10  int arduino_signal [] [] =
    {{8,5,7,9},{7,6,8,9},{7,6,8,9},{7,6,8,9},{7,6,8,9},{6,7,5,9},{7,6,8
    ,9},{7,6,8,9},{7,6,8,9},{5,8,7,9},{7,6,8,9},{7,6,8,9}};
11  int arduino_signal_rev [] [] =
    {{13,10,12,9},{12,11,13,9},{12,11,13,9},{12,11,13,9},{12,11,13,9},{
    11,12,10,9},{12,11,13,9},{12,11,13,9},{12,11,13,9},{10,13,12,9},{12
    ,11,13,9},{12,11,13,9}};
12  int arduino_signal_rev_longer [] [] =
    {{17,14,16,9},{16,15,17,9},{16,15,17,9},{16,15,17,9},{16,15,17,9},{
    15,16,14,9},{16,15,17,9},{16,15,17,9},{16,15,17,9},{14,17,16,9},{16
    ,15,17,9},{16,15,17,9}};
13  //load fonts
14  PFont font;
15  int fontcolor = 0;
16  int fontsize = 24;
17
18  //define tables and arrays to be used for program
19  Table Position;
20  Table Tide_Records;
21  Table Tide_Amp_Data;
22  String[] filenames_M;
23  int ref_Mesocosm_Data_n=0;
24
25  //define H and W of program for various tasks
26  int W = 1100;
27  int H = 900;
28
29  //load PNG of image
30  PImage Mesocosms;
31
32  //Page number that program is on and if it should update.  By defau
    lt set to do nothing (0 for update).  For pages, 1 is main page, 2
    is for individual meso pages
33  int Update_Page = 0;
34  int Page_number = 1;
35  int meso_Page = 0;
36
```

```
37
38      //Various triggers for polling, data conversion, and page updates
39      int Update_time_trigger=1;
40      int trigger_for_triggers=1;
41      int tidal_motion_trigger = 0;
42      int poll_trigger = 1;
43      int poll_trigger_s = 1;
44      int tidal_height_trigger = 1;
45      String[] meso_reading_list;
46      int reading_m_s = 0;
47
48      float R1 [] =
        {4000,4000,4000,4000,4000,4000,4000,4000,4000,4000,4000,4000};
49      float R2 [] = {60,60,60,60,60,60,60,60,60,60,60,60};
50      float offset [] = {30,30,30,30,30,30,30,30,30,30,30,30};
51      float marsh_h_reading [] = {0,0,0,0,0,0,0,0,0,0,0,0};
52
53      //variables for automating tides
54
55
56      int mesocosm_tides_moved_check [] = {0,0,0,0,0,0,0,0,0,0,0,0};
57          //0 is stopped
58          //1 is moving
59      int mesocosm_auto_tides_moved_check [] = {0,0,0,0,0,0,0,0,0,0,0,0};
60        //initial readings from Arduino
61      float mesocosm_surge_water_h_inital [] = {0,0,0,0,0,0,0,0,0,0,0,0};
62      //how far along we are in the tides
63      float mesocosm_increment_height_i =1;
64      //calculate what step we are trying to reach
65      float mesocosm_surge_water_h_step [] = {0,0,0,0,0,0,0,0,0,0,0,0};
66      //check to make sure we have data
67      int initial_surge_reading = 0;
68      //check if we are adding or subtracting.  Need to add to <,> functi
        on
69      float rising_falling_trigger = 1;
70      //how far we are moving tides
71      float incremental_movement = 1;
72      //check if there is new data from the arduino
73      int new_data = 0;
74      //check to make sure data is being written
75      int poll_i_o = 0;
76      //check to see what mesocosm we are on
```

```
77      int meso_number_n=1;
78      int meso_number_i_tide = 0;
79      //check for a small delay in the begining
80      int open_close_valves_i = 0;
81      //variable to open valves to raise (1) or lower (2) tide in marsh t
        ank (0 is nothing)
82      int tide_status = 0;
83      //check to make sure that we are getting data back; reset after 4 t
        ries
84      int reset_check_i = 0;
85
86      float l_tide_amp_today;
87      float h_tide_amp_today;
88      float l_tide_time_today;
89      float h_tide_time_today;
90      float l_tide_dur_today;
91      float h_tide_dur_today;
92
93      float tide_amp_yesterday;
94      float tide_time_yesterday;
95      float tide_dur_yesterday;
96      float tide_amp_tomorrow;
97      float tide_time_tomorrow;
98      float tide_dur_tomorrow;
99
100     float amp_used;
101     float time_used;
102     float time_i;
103     float time_min_quarter;
104
105     float i_tide_amp_today_tm;
106     float ii_tide_amp_today_tm;
107     float l_tide_time_today_tm;
108     float h_tide_time_today_tm;
109     float l_tide_dur_today_tm;
110     float h_tide_dur_today_tm;
111     float l_tide_amp_today_tm;
112     float h_tide_amp_today_tm;
113
114     float dif_tide_amp_today;
115     float dif_tide_time_today;
116     float dif_tide_dur_today;
117
118     float h_tide_start;
119     float h_tide_end;
```

```
120     float l_tide_start;
121     float l_tide_end;
122     float sign_tide_h_l;
123
124
125     String Date_Text = nf(month(),2)+"_"+ nf(day(),2);
126
127     //variables for main page
128     int row = 45;
129     int col = 175;
130     String [] header = {"Mesocosm","Water Height","Projected
        Height","Status","Page Buttons"};
131     int ColorSetBoxFill = 250;
132     int SetBoxColor=0;
133     int YButton_Main_Page = row*4/5;
134     int XButton_Main_Page = 100;
135
136     //variables for meso_tide_set_page
137
138       //Height and Width for Buttons to Appear
139       int Text_Box_X=H/8;
140       int Text_Box_Y=W*5/8;
141       String [] Text_Box_String = {"TIDES","AMP. TIME","AMP.
        HEIGHT","DURATION"};
142
143       //variables for data input
144       int[] box_coding = {0,0};
145       int data_input = 0;
146       String data_input_sting = "";
147       int today_date_int = 0;
148       int date_used;
149       float[] graph_variables = {0,0,0,0,0,0};
150
151     //variables for mesocosm n page (graphs)
152
153       int meso_page_int=0;
154
155       //variables for table size and start location
156       int Graph_Y_TL = 100;
157       int Graph_Y_Dist = 500;
158
```

```
159       int Graph_Y_TL_mp = 200;
160       int Graph_Y_Dist_mp = 500;
161
162       //axis variables
163       String Times[] = {"12 AM","6 AM","12 PM","6 PM","12 AM"};
164       String Depths[] = {"30 cm","23 cm","15 cm","8 cm","Marsh","-8
        cm","-15 cm","-23 cm","-30 cm"};
165       int Graph_X_Side_Bar = 125;
166       int Graph_X_TL = Graph_X_Side_Bar;
167       int Graph_X_Dist = W-Graph_X_Side_Bar*2;
168       int Graph_X_BR = Graph_X_TL+Graph_X_Dist;
169       int Graph_Y_BR = Graph_Y_TL+Graph_Y_Dist;
170       int Graph_Y_BR_mp = Graph_Y_TL_mp+Graph_Y_Dist_mp;
171       float y_marsh;
172       float y_marsh_sum;
173       float marsh_sum;
174       int count =0;
175
176         //X Axis Markers
177         int X_Axis_X = 8;
178         int X_Axis_Y = 5;
179
180         //Y Axis Rectangles
181         int Y_Axis_X = 5;
182         int Y_Axis_Y = 15;
183
184        //Left and Right Arrow Options
185       int Arrow_LR_X_Dist = 50;
186       int Arrow_LR_Y_Dist = 50;
187       int Arrow_LR_Meso_X_Loc_1 = W/12;
188       int Arrow_LR_Meso_Y_Loc_1 = Graph_Y_TL/2+H*3/4;
189       int Arrow_LR_Meso_X_Loc_1_mp = W/3;
190       int Arrow_LR_Meso_Y_Loc_1_mp = Arrow_LR_Y_Dist/2;
191       int Arrow_LR_Meso_X_Loc_2 = W/3;
192       int Arrow_LR_Meso_Y_Loc_2 = Graph_Y_TL*1/4;
193       int Arrow_LR_Meso_Y_Loc_2_mp = Graph_Y_TL_mp*1/4;
194
195       //individual meso information location
196       float reading_m;
```

```
197     float meso_reading_now;
198     float tide_pattern_now;
199     int ind_meso_information = Graph_Y_TL_mp/2+H*3/4;
200
201     //Arrow Images
202     PImage Arrow;
203     PImage Arrow_LR;
204
205     //table title information
206
207     //Manual operation page variables
208     int backgroundcolor=150;
209       //start position of row
210     int XRow1 = 50;
211     int YRow1 = 350;
212       //spacing of rows
213     int Yspace= 40;
214       //size of Flow control buttons
215     int XASButton = 125;
216     int YASButton = 35;
217       //size of set buttons
218     int XSButton = 50;
219     int YSButton = 35;
220       //location of control text
221     int XrefText=XASButton/2;
222     int YrefText=YASButton/2;
223       //location of set text
224     int XSrefText=XSButton/2;
225     int YSrefText=YSButton/2;
226       //color of Flow control button
227     int colorASButton=200;
228       //Color of Mesocosm Flow Status
229     int colorSButton = 250;
230       //selection variables
231     int selection_flow1=0;
232     int[] tide_status1={2,2,2,2,2,2,2,2,2,2,2,2,2};
233     String[] tide_status_selection = {"Raise Tide","Lower Tide","Stop
    Tide","Null"};
234     String[] tide_auto_manual = {"Auto","Manual"};
235     int Meso = 0;
236     int[] MesoRow = {0,0,0,0,0,0,0,0,0,0,0,0,0};
237     float tide_difference;
```

```
238       float tide_stop_difference;
239
240       //check for next day
241       int next_day_check=0;
242       String day_check;
243
244     void setup() {
245       //define size of program
246       size(1100, 900);
247       background(209);
248
249       //load font
250       font = loadFont("TimesNewRomanPSMT-48.vlw");
251       fill(fontcolor);
252       textFont(font, fontsize);
253       textAlign(CENTER,CENTER);
254
255       //load images
256       Mesocosms = loadImage("Meso_Number.png");
257       imageMode(CENTER);
258       Arrow = loadImage("Arrow.png");
259       Arrow_LR = loadImage("Arrow_LR.png");
260
261       // print list of available serial ports and set it
262       println(Serial.list());
263       xbee = new Serial(this, Serial.list()[1], 9600);
264
265       //check to see if a file for <today> needs to be created
266       file_check();
267       Tide_Records = loadTable("Mesocosm_data_for_"+nf(month(),2)+"_"+n
    f(day(),2)+"_"+nf(year(),4)+".csv", "header, csv");
268
269       //check to see what mesocosm pages have been created
270       filenames_M = listFileNames_M(sketchPath("data"));
271
272       //load main page
273       main_page();
274
275       //load variables from various pages
276
277       i_tide_amp_today_tm = Tide_Amp_Data.getFloat(today_date_int,2);
```

```
278       ii_tide_amp_today_tm= Tide_Amp_Data.getFloat(today_date_int,5);
279
280         if(i_tide_amp_today_tm<ii_tide_amp_today_tm){
281           l_tide_time_today_tm = 3600*Tide_Amp_Data.getFloat(today_date
      _int,1);
282           h_tide_time_today_tm = 3600*Tide_Amp_Data.getFloat(today_date
      _int,4);
283           l_tide_dur_today_tm = 3600*Tide_Amp_Data.getFloat(today_date_
      int,3);
284           h_tide_dur_today_tm = 3600*Tide_Amp_Data.getFloat(today_date_
      int,6);
285           l_tide_amp_today_tm = i_tide_amp_today_tm;
286           h_tide_amp_today_tm = ii_tide_amp_today_tm;
287         } else {
288           h_tide_time_today_tm = 3600*Tide_Amp_Data.getFloat(today_date
      _int,1);
289           l_tide_time_today_tm = 3600*Tide_Amp_Data.getFloat(today_date
      _int,4);
290           h_tide_dur_today_tm = 3600*Tide_Amp_Data.getFloat(today_date_
      int,3);
291           l_tide_dur_today_tm = 3600*Tide_Amp_Data.getFloat(today_date_
      int,6);
292           h_tide_amp_today_tm = i_tide_amp_today_tm;
293           l_tide_amp_today_tm = ii_tide_amp_today_tm;
294         }
295     }
296
297     void draw(){
298
299       //reset updates triggers for screen updates, mesocosm polling,
300       if(int(second()+1) % 55 == 0 && trigger_for_triggers==1){
301         Update_time_trigger=1;
302         trigger_for_triggers=0;
303       }
304       //reset Arduino Uno if data is not coming in
305       if(reset_check_i>2){
306         xbee.stop();
307         xbee = new Serial(this, Serial.list()[1], 9600);
308         reset_check_i=0;
309         println("Ard. Reset");
310         println(reset_check_i);
```

```
311       }
312
313       //15 second trigger
314       //check water height of mesocosms
315       if(second() % 15 == 0 && poll_trigger==1){
316         poll_i_o=1;
317         poll_trigger=0;
318         reading_m_s = 2;
319         xbee.write("<0,9,>");
320     //    println("Sent");
321         reset_check_i=reset_check_i+1;
322       }
323
324       if((second()+6) % 15 == 0 && poll_trigger_s==1){
325         poll_i_o=1;
326         poll_trigger_s=0;
327         reading_m_s = 1;
328         xbee.write("<0,0,>");
329         println("Sent_1");
330         reset_check_i=reset_check_i+1;
331       }
332
333       if((second()+1) % 15 == 0 && poll_trigger==0){
334     //    println("ok?");
335         poll_trigger=1;
336         poll_trigger_s=1;
337
338
339       }
340
341       tidal_motion();
342
343       if(second()==30 && tidal_motion_trigger==1){
344         tidal_motion_trigger=0;
345       }
346       if(second()==11 && tidal_motion_trigger==0){
347         meso_number_n=1;
348         tidal_motion_trigger=1;
349       }
350
```

```
351       //check if update is needed
352       if(Page_number == 1 && Update_Page==1){
353         //Display Main Page and Turn off Update
354         main_page();
355         Update_Page=0;
356       } else if(Page_number == 2 && Update_Page==1){
357         individual_meso_page();
358         Update_Page=0;
359       }  else if(Page_number == 2 && Update_Page==1 &&
      ref_Mesocosm_Data_n==meso_page_int-1){
360         individual_meso_page();
361         Update_Page=0;
362       } else if(Page_number == 3 && Update_Page==1){
363         meso_tide_set_page();
364         Update_Page=0;
365       } else if(Page_number == 4 && Update_Page==1){
366         manual_override();
367         Update_Page=0;
368       }
369
370       //Update Main Page with new data every minute (when second on
      clock =59)
371       if((Page_number == 1 || Page_number == 2)&& int(second()+1) % 59
      == 0 && Update_time_trigger==1){
372         textAlign(LEFT,BOTTOM);
373         rectMode(CORNER);
374         fill(209);
375         noStroke();
376         rect(9*width/11,height-20,2*width/11,20);
377         stroke(0);
378         fill(0);
379         text("Updated
      "+nf(hour(),2)+":"+nf(minute()+1,2),9*width/11,height);
380         Update_time_trigger=0;
381         trigger_for_triggers=1;
382         Update_Page=1;
383         if(meso_page_int!=ref_Mesocosm_Data_n+1){
384           Update_Page=0;
```

```
385         }
386       }
387
388       //check to see if a page needs to be created for the next day
389       if(hour()==22 && next_day_check==0){
390       int [] days_month = {31,28,31,30,31,30,31,31,30,31,30,31};
391         if(day()==days_month[int(month()-1)]){
392           day_check=nf(month()+1,2)+"_"+"01";
393         }else{
394           day_check=nf(month(),2)+"_"+nf(int(day()+1),2);
395         }
396
397         if(day()==31 && month() == 12){
398               day_check="01_01";
399            }
400         if(month()==2 && day()==28 && year()%4==0){
401           day_check="02_29";
402         }
403         if(month()==2 && day()==29){
404           day_check="03_01";
405         }
406
407         file_check_named(day_check);
408         next_day_check=1;
409       }
410
411       if(hour()==1 && next_day_check==1){
412         next_day_check=0;
413       }
414
415     }
416
417     //check for files in data directory
418     String[] listFileNames(String dir) {
419       File file = new File(dir);
420       if (file.isDirectory()) {
421         String names[] = file.list();
422         return names;
423       } else {
424         // If it&apos;s not a directory
425         return null;
```

```
426       }
427     }
428
429     String[] listFileNames_M(String dir) {
430       File file = new File(dir);
431       String [] stored_files = {};
432       if (file.isDirectory()) {
433         String names[] = file.list();
434
435         for(int i=0;i<names.length;i++){
436           String filename = names[i];
437           if(filename.contains("Mesocosm")){
438             meso_page_int=meso_page_int+1;
439             stored_files = append(stored_files,names[i].substring(18,28
    ).replace(&apos;_&apos;,&apos;-&apos;));
440             if(names[i].equals("Mesocosm_data_for_"+nf(month(),2)+"_"+n
    f(day(),2)+"_"+nf(year(),4)+".csv")){
441               ref_Mesocosm_Data_n=meso_page_int-1;
442               println(ref_Mesocosm_Data_n);
443               println("---");
444               println(stored_files);
445
446             }
447           }
448         }
449         return stored_files;
450       } else {
451         // If it&apos;s not a directory
452         return null;
453       }
454     }
455
456     void serialEvent(Serial xbee){
457
458       while (xbee.available() > 0) {
459         myString = xbee.readString();
460         if (myString != null ) {//check for number here //<>//
461
           myStrings=myStrings+myString.replace(&apos;\n&apos;,&apos;
    &apos;).replace(&apos;\r&apos;,&apos; &apos;);
462           myStrings=myStrings.trim();
```

```
463         }
464         if (myString.equals("S")) {
465           poll_i_o=0;
466           println("S");
467           meso_number_n=meso_number_n+1;
468         }
469         if (myString.equals("F")) {
470           println("F");
471
          mesocosm_tides_moved_check[meso_number_n-
      1]=abs(abs(mesocosm_tides_moved_check[meso_number_n-1])+int(pow(-
      1,float(mesocosm_tides_moved_check[meso_number_n-1]))));
472           poll_i_o=0;
473         }
474         if (myString.equals("!")) {
475           println("err: "+myStrings);
476           myStrings = "";
477         }
478         if (myString.equals(";")) {
479           myStrings=myStrings.replace(&apos;S&apos;,&apos; &apos;);
480           myStrings=myStrings.trim();
481           println("rec: "+myStrings);
482           reset_check_i=0;
483           record_heights();
484           myStrings = "";
485         }
486       }
487       xbee.clear();
488     }
489
490
491     void mousePressed(){
492
493     //  meso_number_n=1;
494       //page 1 is main page
495       //page 2 is individual meso page
496       //page 3 is set page
497       //page 4 is override page
498
499       if(Page_number == 1){
500         for(int i=1;i<13;i++){
```

```
501
        if(Update_Page==0 && mouseY>=row*i+height/3 &&
      mouseY<=row*i+height/3+YButton_Main_Page && mouseX>= col*4+width/5
      && mouseX<= col*4+width/5+XButton_Main_Page){
502           meso_Page=i;
503           Page_number=2;
504           Update_Page=1;
505         }
506       }
507       if(Update_Page==0 && mouseX>= W*7/8 && mouseX<=
      W*7/8+XButton_Main_Page && mouseY>=row &&
      mouseY<=row+YButton_Main_Page){
508         Page_number=3;
509         Update_Page=1;
510       }
511       if(Update_Page==0 && mouseX>= W*7/8 && mouseX<=
      W*7/8+XButton_Main_Page && mouseY>=row*2 &&
      mouseY<=row*2+YButton_Main_Page){
512         Page_number=4;
513         Update_Page=1;
514       }
515     }
516
517      if(Page_number == 2){
518
519        if(Update_Page==0 && mouseX>= W*7/8 && mouseX<=
      W*7/8+XButton_Main_Page && mouseY>=row &&
      mouseY<=row+YButton_Main_Page){
520           Page_number=1;
521           Update_Page=1;
522         }
523
524        if(Update_Page==0 && mouseX>= W*7/8 && mouseX<=
      W*7/8+XButton_Main_Page && mouseY>=row*2 &&
      mouseY<=row*2+YButton_Main_Page){
525           Page_number=3;
526           Update_Page=1;
527         }
528
```

```
529
        if(Update_Page==0 && mouseX>= W*7/8 && mouseX<=
      W*7/8+XButton_Main_Page && mouseY>=row*3 &&
      mouseY<=row*3+YButton_Main_Page){
530       Page_number=4;
531       Update_Page=1;
532     }
533
534
        if(mouseX>= Arrow_LR_Meso_X_Loc_1_mp-Arrow_LR_X_Dist/2 &&
      mouseX<= Arrow_LR_Meso_X_Loc_1_mp+Arrow_LR_X_Dist/2 &&
      mouseY>=Arrow_LR_Meso_Y_Loc_1_mp-Arrow_LR_Y_Dist/2+1 &&
      mouseY<=Arrow_LR_Meso_Y_Loc_1_mp+Arrow_LR_Y_Dist/2+1){
535       if(1==meso_Page){
536         meso_Page=12;
537         individual_meso_page();
538       }else{
539         meso_Page=meso_Page-1;
540         individual_meso_page();
541       }
542     }
543
544
        if(mouseX>= Arrow_LR_Meso_X_Loc_2+Arrow_LR_Meso_X_Loc_1_mp-
      Arrow_LR_X_Dist/2 && mouseX<=
      Arrow_LR_Meso_X_Loc_2+Arrow_LR_Meso_X_Loc_1_mp+Arrow_LR_X_Dist/2 &&
      mouseY>=Arrow_LR_Meso_Y_Loc_1_mp-Arrow_LR_Y_Dist/2 &&
      mouseY<=Arrow_LR_Meso_Y_Loc_1_mp+Arrow_LR_Y_Dist/2){
545       if(12==meso_Page){
546         meso_Page=1;
547         individual_meso_page();
548       }else{
549         meso_Page=meso_Page+1;
550         individual_meso_page();
551       }
552     }
553
```

```
554
        if(mouseX>= Arrow_LR_Meso_X_Loc_1_mp-Arrow_LR_X_Dist/2 &&
      mouseX<= Arrow_LR_Meso_X_Loc_1_mp+Arrow_LR_X_Dist/2 &&
      mouseY>=Arrow_LR_Meso_Y_Loc_1_mp+Arrow_LR_Y_Dist*3/2 &&
      mouseY<=Arrow_LR_Meso_Y_Loc_1_mp+Arrow_LR_Y_Dist*5/2){


555           if(0==ref_Mesocosm_Data_n){
556             ref_Mesocosm_Data_n=meso_page_int-1;
557             Tide_Records = loadTable("Mesocosm_data_for_"+filenames_M
      [ref_Mesocosm_Data_n].replace(&apos;-&apos;,&apos;_&apos;)+".csv");
558             individual_meso_page();
559           }else{
560             ref_Mesocosm_Data_n=ref_Mesocosm_Data_n-1;
561             Tide_Records = loadTable("Mesocosm_data_for_"+filenames_M
      [ref_Mesocosm_Data_n].replace(&apos;-&apos;,&apos;_&apos;)+".csv");
562             individual_meso_page();
563           }
564         }
565
566


        if(mouseX>= Arrow_LR_Meso_X_Loc_2+Arrow_LR_Meso_X_Loc_1_mp-
      Arrow_LR_X_Dist/2 && mouseX<=
      Arrow_LR_Meso_X_Loc_2+Arrow_LR_Meso_X_Loc_1_mp+Arrow_LR_X_Dist/2 &&
      mouseY>=Arrow_LR_Meso_Y_Loc_1_mp+Arrow_LR_Y_Dist*3/2 &&
      mouseY<=Arrow_LR_Meso_Y_Loc_1_mp+Arrow_LR_Y_Dist*5/2){


567           if(meso_page_int==ref_Mesocosm_Data_n+1){
568             ref_Mesocosm_Data_n=0;
569             Tide_Records = loadTable("Mesocosm_data_for_"+filenames_M
      [ref_Mesocosm_Data_n].replace(&apos;-&apos;,&apos;_&apos;)+".csv");
570             individual_meso_page();
571           }else{
572             ref_Mesocosm_Data_n=ref_Mesocosm_Data_n+1;
```

```
573                 Tide_Records = loadTable("Mesocosm_data_for_"+filenames_M
    [ref_Mesocosm_Data_n].replace(&apos;-&apos;,&apos;_&apos;)+".csv");
574                 individual_meso_page();
575               }
576             }
577             Tide_Records = loadTable("Mesocosm_data_for_"+nf(month(),2)+"
    _"+nf(day(),2)+"_"+nf(year(),4)+".csv", "header, csv");
578           }
579
580         if(Page_number == 3){
581
582
        if(mouseX>=Text_Box_X+col*4-XButton_Main_Page/2 && mouseX <=
    Text_Box_X+col*4-XButton_Main_Page/2+XButton_Main_Page && mouseY>=
    Text_Box_Y+row*2-YButton_Main_Page/2 && mouseY<= Text_Box_Y+row*2-
    YButton_Main_Page/2+YButton_Main_Page){


583             //load variables
584
585             float i_tide_amp_today =
    Tide_Amp_Data.getFloat(today_date_int,2);
586             float ii_tide_amp_today =
    Tide_Amp_Data.getFloat(today_date_int,5);
587
588             if(i_tide_amp_today<=ii_tide_amp_today){
589               l_tide_amp_today = Tide_Amp_Data.getFloat(date_used,2);
590               h_tide_amp_today = Tide_Amp_Data.getFloat(date_used,5);
591               l_tide_time_today = Tide_Amp_Data.getFloat(date_used,1);
592               h_tide_time_today = Tide_Amp_Data.getFloat(date_used,4);
593               l_tide_dur_today = Tide_Amp_Data.getFloat(date_used,3);
594               h_tide_dur_today = Tide_Amp_Data.getFloat(date_used,6);
595             }else{
596               l_tide_amp_today = Tide_Amp_Data.getFloat(date_used,5);
597               h_tide_amp_today = Tide_Amp_Data.getFloat(date_used,2);
```

```
598             l_tide_time_today = Tide_Amp_Data.getFloat(date_used,4);
599             h_tide_time_today = Tide_Amp_Data.getFloat(date_used,1);
600             l_tide_dur_today = Tide_Amp_Data.getFloat(date_used,6);
601             h_tide_dur_today = Tide_Amp_Data.getFloat(date_used,3);
602           }
603           //set variables for week
604
605           for(int i=1;i<8;i++){
606             Tide_Amp_Data.setFloat(date_used+i,2,l_tide_amp_today);
607             Tide_Amp_Data.setFloat(date_used+i,5,h_tide_amp_today);
608             Tide_Amp_Data.setFloat(date_used+i,1,l_tide_time_today);
609             Tide_Amp_Data.setFloat(date_used+i,4,h_tide_time_today);
610             Tide_Amp_Data.setFloat(date_used+i,3,l_tide_dur_today);
611             Tide_Amp_Data.setFloat(date_used+i,6,h_tide_dur_today);
612           }
613           cursor(WAIT);
614           saveTable(Tide_Amp_Data, "data/Tide_Amp_Data.csv");
615           cursor(ARROW);
616         }
617
618
```

```
    if(mouseX>=Text_Box_X+col*4-XButton_Main_Page/2 && mouseX <=
Text_Box_X+col*4-XButton_Main_Page/2+XButton_Main_Page && mouseY>=
Text_Box_Y+row*3-YButton_Main_Page/2 && mouseY<= Text_Box_Y+row*3-
YButton_Main_Page/2+YButton_Main_Page){
```

```
619           //load variables
620
621           float i_tide_amp_today =
Tide_Amp_Data.getFloat(today_date_int,2);
622           float ii_tide_amp_today=
Tide_Amp_Data.getFloat(today_date_int,5);
623
624           if(i_tide_amp_today<=ii_tide_amp_today){
```

```
625             l_tide_amp_today = Tide_Amp_Data.getFloat(date_used,2);
626             h_tide_amp_today = Tide_Amp_Data.getFloat(date_used,5);
627             l_tide_time_today = Tide_Amp_Data.getFloat(date_used,1);
628             h_tide_time_today = Tide_Amp_Data.getFloat(date_used,4);
629             l_tide_dur_today = Tide_Amp_Data.getFloat(date_used,3);
630             h_tide_dur_today = Tide_Amp_Data.getFloat(date_used,6);
631           }else{
632             l_tide_amp_today = Tide_Amp_Data.getFloat(date_used,5);
633             h_tide_amp_today = Tide_Amp_Data.getFloat(date_used,2);
634             l_tide_time_today = Tide_Amp_Data.getFloat(date_used,4);
635             h_tide_time_today = Tide_Amp_Data.getFloat(date_used,1);
636             l_tide_dur_today = Tide_Amp_Data.getFloat(date_used,6);
637             h_tide_dur_today = Tide_Amp_Data.getFloat(date_used,3);
638           }
639
640
641
642           //set variables for week
643
644           for(int i=1;i<31;i++){
645             Tide_Amp_Data.setFloat(date_used+i,2,l_tide_amp_today);
646             Tide_Amp_Data.setFloat(date_used+i,5,h_tide_amp_today);
647             Tide_Amp_Data.setFloat(date_used+i,1,l_tide_time_today);
648             Tide_Amp_Data.setFloat(date_used+i,4,h_tide_time_today);
649             Tide_Amp_Data.setFloat(date_used+i,3,l_tide_dur_today);
650             Tide_Amp_Data.setFloat(date_used+i,6,h_tide_dur_today);
651           }
652           cursor(WAIT);
653           saveTable(Tide_Amp_Data, "data/Tide_Amp_Data.csv");
```

```
654           cursor(ARROW);
655         }
656
657
          if(mouseX>=Text_Box_X+col*5-XButton_Main_Page/2 && mouseX <=
        Text_Box_X+col*5-XButton_Main_Page/2+XButton_Main_Page && mouseY>=
        Text_Box_Y+row-YButton_Main_Page/2 && mouseY<= Text_Box_Y+row-
        YButton_Main_Page/2+YButton_Main_Page){
658           date_used=date_used-1;
659           meso_tide_set_page();
660         }
661
662
          if(mouseX>=Text_Box_X+col*5-XButton_Main_Page/2 && mouseX <=
        Text_Box_X+col*5-XButton_Main_Page/2+XButton_Main_Page && mouseY>=
        Text_Box_Y+row*2-YButton_Main_Page/2 && mouseY<= Text_Box_Y+row*2-
        YButton_Main_Page/2+YButton_Main_Page){
663           date_used=date_used+1;
664           meso_tide_set_page();
665         }
666
667
          if(mouseX>=Text_Box_X+col*4-XButton_Main_Page/2 && mouseX <=
        Text_Box_X+col*4-XButton_Main_Page/2+XButton_Main_Page && mouseY>=
        Text_Box_Y+row-YButton_Main_Page/2 && mouseY<= Text_Box_Y+row-
        YButton_Main_Page/2+YButton_Main_Page){
668           cursor(WAIT);
669
670           //load variables for week
671
672           float i_tide_amp_today =
        Tide_Amp_Data.getFloat(today_date_int,2);
673           float ii_tide_amp_today =
        Tide_Amp_Data.getFloat(today_date_int,5);
674
675           if(i_tide_amp_today<=ii_tide_amp_today){
676             l_tide_amp_today = Tide_Amp_Data.getFloat(date_used,2);
677             h_tide_amp_today = Tide_Amp_Data.getFloat(date_used,5);
```

```
678             l_tide_time_today = Tide_Amp_Data.getFloat(date_used,1);
679             h_tide_time_today = Tide_Amp_Data.getFloat(date_used,4);
680             l_tide_dur_today = Tide_Amp_Data.getFloat(date_used,3);
681             h_tide_dur_today = Tide_Amp_Data.getFloat(date_used,6);
682           }else{
683             l_tide_amp_today = Tide_Amp_Data.getFloat(date_used,5);
684             h_tide_amp_today = Tide_Amp_Data.getFloat(date_used,2);
685             l_tide_time_today = Tide_Amp_Data.getFloat(date_used,4);
686             h_tide_time_today = Tide_Amp_Data.getFloat(date_used,1);
687             l_tide_dur_today = Tide_Amp_Data.getFloat(date_used,6);
688             h_tide_dur_today = Tide_Amp_Data.getFloat(date_used,3);
689           }
690
691           i_tide_amp_today_tm = Tide_Amp_Data.getFloat(today_date_int,2
    );
692           ii_tide_amp_today_tm = Tide_Amp_Data.getFloat(today_date_int,
    5);
693
694           if(i_tide_amp_today_tm<ii_tide_amp_today_tm){
695             l_tide_time_today_tm = 3600*Tide_Amp_Data.getFloat(today_da
    te_int,1);
696             h_tide_time_today_tm = 3600*Tide_Amp_Data.getFloat(today_da
    te_int,4);
697             l_tide_dur_today_tm = 3600*Tide_Amp_Data.getFloat(today_dat
    e_int,3);
698             h_tide_dur_today_tm = 3600*Tide_Amp_Data.getFloat(today_dat
    e_int,6);
699             l_tide_amp_today_tm = i_tide_amp_today_tm;
700             h_tide_amp_today_tm = ii_tide_amp_today_tm;
701           } else {
702             h_tide_time_today_tm = 3600*Tide_Amp_Data.getFloat(today_da
    te_int,1);
703             l_tide_time_today_tm = 3600*Tide_Amp_Data.getFloat(today_da
    te_int,4);
704             h_tide_dur_today_tm = 3600*Tide_Amp_Data.getFloat(today_dat
    e_int,3);
```

```
705             l_tide_dur_today_tm = 3600*Tide_Amp_Data.getFloat(today_dat
    e_int,6);
706             h_tide_amp_today_tm = i_tide_amp_today_tm;
707             l_tide_amp_today_tm = ii_tide_amp_today_tm;
708           }
709
710           h_tide_start = h_tide_time_today-h_tide_dur_today/2;
711           h_tide_end = h_tide_time_today+h_tide_dur_today/2;
712           l_tide_start = l_tide_time_today-l_tide_dur_today/2;
713           l_tide_end = l_tide_time_today+l_tide_dur_today/2;
714
715
716           for(int i=0;i<5760;i=i+1){
717             if(i>=int(4*60*h_tide_start) && i<=int(4*60*h_tide_end)){
718               Tide_Records.setFloat(i ,3, h_tide_amp_today);
719             }
720             if(i>=int(4*60*l_tide_start) && i<=int(4*60*l_tide_end)){
721               Tide_Records.setFloat(i ,3, l_tide_amp_today);
722             }
723             if(i>int(4*60*h_tide_end) && i<int(4*60*(l_tide_start))){
724               Tide_Records.setFloat(i ,3, h_tide_amp_today-
    ((floor(float(i)/5)*5-(4*60*h_tide_end))/(4*60*(l_tide_start-
    h_tide_end)))*(h_tide_amp_today-l_tide_amp_today));
725             }
726             if(i>int(4*60*l_tide_end) && i<int(4*60*(h_tide_start))){
727               Tide_Records.setFloat(i ,3, l_tide_amp_today-
    ((floor(float(i)/5)*5-(4*60*l_tide_end))/(4*60*(h_tide_start-
    l_tide_end)))*(l_tide_amp_today-h_tide_amp_today));
728             }
729
730             if(i<int(4*60*l_tide_start) && i<int(4*60*(h_tide_start))){
```

```
731                 if(l_tide_time_today<h_tide_time_today){
732                   amp_used = l_tide_amp_today;
733                   sign_tide_h_l = (h_tide_time_today-
    l_tide_time_today)/(abs(h_tide_time_today-l_tide_time_today));
734                 }else{
735                   amp_used = h_tide_amp_today;
736                   sign_tide_h_l = (h_tide_time_today-
    l_tide_time_today)/(abs(h_tide_time_today-l_tide_time_today));
737                 }
738                 Tide_Records.setFloat(i ,3,
    amp_used+sign_tide_h_l*((floor(float(i)/5)*5-
    (4*60*min(l_tide_start,h_tide_start)))/(4*60*(min(abs(h_tide_start-
    l_tide_end),abs(l_tide_start-h_tide_end)))))*(l_tide_amp_today-
    h_tide_amp_today));
739               }
740               if(i>int(4*60*l_tide_end) && i>int(4*60*(h_tide_end))){
741                 if(l_tide_time_today>h_tide_time_today){
742                   amp_used = l_tide_amp_today;
743                   sign_tide_h_l = (h_tide_time_today-
    l_tide_time_today)/(abs(h_tide_time_today-l_tide_time_today));
744                 }else{
745                   amp_used = h_tide_amp_today;
746                   sign_tide_h_l = (h_tide_time_today-
    l_tide_time_today)/(abs(h_tide_time_today-l_tide_time_today));
747                 }
748
                  Tide_Records.setFloat(i ,3,
    amp_used+sign_tide_h_l*((floor(float(i)/5)*5-
    (4*60*max(l_tide_end,h_tide_end)))/(4*60*(min(abs(h_tide_start-
    l_tide_end),abs(l_tide_start-h_tide_end)))))*(l_tide_amp_today-
    h_tide_amp_today));

749               }
750             }
751             saveTable(Tide_Records,
    "data/Mesocosm_data_for_"+Date_Text+"_"+nf(year(),4)+".csv");
752             cursor(ARROW);
753           }
```

```
754
755         if(Update_Page==0 && mouseX>= W*7/8 && mouseX<=
        W*7/8+XButton_Main_Page && mouseY>=row &&
        mouseY<row+YButton_Main_Page){
756           Page_number=1;
757           Update_Page=1;
758         }
759         if(data_input ==0){
760           for(int i = 1;i<4;i++){
761             for(int ii = 1;ii<3;ii++){
762
                 if(mouseX>=Text_Box_X+col*i-XButton_Main_Page/2 &&
        mouseY>=Text_Box_Y+row*ii-YButton_Main_Page/2 &&
763
                    mouseX<=Text_Box_X+col*i-
        XButton_Main_Page/2+XButton_Main_Page && mouseY<=Text_Box_Y+row*ii-
        YButton_Main_Page/2+YButton_Main_Page){
764                     fill(100);
765                     box_coding [0]=i;
766                     box_coding [1]=ii;
767
                        rect(Text_Box_X+col*i-
        XButton_Main_Page/2,Text_Box_Y+row*ii-
        YButton_Main_Page/2,XButton_Main_Page,YButton_Main_Page);
768                     data_input = 1;
769                     data_input_sting="";
770                   }
771             }
772           }
773         }
774       }
775
776       if(Page_number == 4){
777
778         if(Update_Page==0 && mouseX>= W*7/8 && mouseX<=
        W*7/8+XButton_Main_Page && mouseY>=row &&
        mouseY<=row+YButton_Main_Page){
779           Page_number=1;
780           Update_Page=1;
781         }
782
783         for(int ii=0; ii<12;ii++){
784
```

```
785
        if(mouseX>=XRow1+(XASButton+XASButton/4) &&
      mouseX<=XRow1+(XASButton+XASButton/4)+XASButton && mouseY>=
      YRow1+0+Yspace*ii && mouseY<=YRow1+0+YASButton+Yspace*ii){
786       selection_flow1=1;
787       MesoRow[ii]=1;
788       manual_override();
789       }
790
        if(mouseX>=XRow1+(XASButton+XASButton/4)*2 &&
      mouseX<=XRow1+(XASButton+XASButton/4)*2+XASButton && mouseY>=
      YRow1+0+Yspace*ii && mouseY<=YRow1+0+YASButton+Yspace*ii){
791       selection_flow1=2;
792       MesoRow[ii]=2;
793       manual_override();
794       }
795
        if(mouseX>=XRow1+(XASButton+XASButton/4)*3 &&
      mouseX<=XRow1+(XASButton+XASButton/4)*3+XASButton && mouseY>=
      YRow1+0+Yspace*ii && mouseY<=YRow1+0+YASButton+Yspace*ii){
796       selection_flow1=3;
797       MesoRow[ii]=3;
798       manual_override();
799       }
800
        if(mouseX>=XRow1+(XASButton+XASButton/4)*4 &&
      mouseX<=XRow1+(XASButton+XASButton/4)*4+XASButton && mouseY>=
      YRow1+0+Yspace*ii && mouseY<=YRow1+0+YASButton+Yspace*ii){
801       selection_flow1=4;
802       if(Position.getInt(ii,2)==0){
803         Position.setInt(ii,2,1);
804       }else if(Position.getInt(ii,2)==1){
805         Position.setInt(ii,2,0);
806       }
807       saveTable(Position, "data/PositionData.csv");
808       manual_override();
809       }
810
        if(mouseX>=XRow1+(XASButton+XASButton/4)*5 &&
      mouseX<=XRow1+(XASButton+XASButton/4)*5+XSButton && mouseY>=
      YRow1+0+Yspace*ii && mouseY<=YRow1+0+YSButton+Yspace*ii){
```

```
811           if (MesoRow[ii] != 0){
812           Position.setInt(ii,1,MesoRow[ii]-1);
813           saveTable(Position, "data/PositionData.csv");
814         //  tide_status1[ii]=MesoRow[ii]-1;
815         //  tide_status1=selection_flow1-1;
816           println("Sig"+ii+1+","+arduino_signal[ii][MesoRow[ii]-
      1]+",");
817           println("M"+int(MesoRow[ii]-1));
818           int number=ii+1;
819           println("<"+number+","+arduino_signal[ii][MesoRow[ii]-
      1]+",>");
820           xbee.write("<"+number+","+arduino_signal[ii][MesoRow[ii]-
      1]+",>");
821           }
822           MesoRow[ii]=5;
823           selection_flow1=5;
824           manual_override();
825           manual_override();
826           }
827
828     }
829
830       }
831
832       //for debugging
833       println("X:"+mouseX+";"+width);
834       println("Y:"+mouseY+";"+height);
835     }
836
837     void keyReleased() {
838       if(data_input == 1 && Page_number == 3  && (key!=ENTER &&
      key!=BACKSPACE)){
839         data_input_sting=data_input_sting+key;
840         fill(100);
841
          rect(Text_Box_X+col*box_coding [0]-
      XButton_Main_Page/2,Text_Box_Y+row*box_coding [1]-
      YButton_Main_Page/2,XButton_Main_Page,YButton_Main_Page);
842         textAlign(CENTER, CENTER);
843         fill(250);
844         text(data_input_sting,Text_Box_X+col*box_coding
      [0],Text_Box_Y+row*box_coding [1]);
845       }
```

```
846
847       if(key==ENTER && data_input == 1 && Page_number == 3){
848
849         if(Float.isNaN(float(data_input_sting))){
850           textAlign(CENTER, LEFT);
851           fill(250,0,0);
852           text("Error Not a Number",Text_Box_X+col*4,Text_Box_Y+row);
853
854       } else if(box_coding [0]==1 && (float(data_input_sting)>=24 ||
      float(data_input_sting)<=0)){
855           textAlign(CENTER, LEFT);
856           fill(250,0,0);
857           text("Number too
      Large/Small",Text_Box_X+col*4,Text_Box_Y+row);
858
859       } else if(box_coding [0]==2 && (float(data_input_sting)>=30 ||
      float(data_input_sting)<=-30)){
860           textAlign(CENTER, LEFT);
861           fill(250,0,0);
862           text("Number too
      Large/Small",Text_Box_X+col*4,Text_Box_Y+row);
863
864       } else if(box_coding [0]==3 && (float(data_input_sting)>=24 ||
      float(data_input_sting)<=0)){
865           textAlign(CENTER, LEFT);
866           fill(250,0,0);
867           text("Number too
      Large/Small",Text_Box_X+col*4,Text_Box_Y+row);
868       } else {
869           textAlign(CENTER, CENTER);
870           Tide_Amp_Data.setFloat(today_date_int,box_coding
      [0]+((box_coding [1]-1)*3),float(data_input_sting));
871           saveTable(Tide_Amp_Data, "data/Tide_Amp_Data.csv");
872           meso_tide_set_page();
873       }
874
```

```
875         fill(250);
876
            rect(Text_Box_X+col*box_coding [0]-
        XButton_Main_Page/2,Text_Box_Y+row*box_coding [1]-
        YButton_Main_Page/2,XButton_Main_Page,YButton_Main_Page);

877         data_input = 0;
878         fill(0);
879         textAlign(CENTER, CENTER);
880         text(nf(float(data_input_sting),2,1),Text_Box_X+col*box_coding
        [0],Text_Box_Y+row*box_coding [1]);

881
882       }
883
884       if(key==BACKSPACE && data_input == 1 && Page_number == 3){
885         fill(250);
886
            rect(Text_Box_X+col*box_coding [0]-
        XButton_Main_Page/2,Text_Box_Y+row*box_coding [1]-
        YButton_Main_Page/2,XButton_Main_Page,YButton_Main_Page);

887         data_input = 0;
888       }
889
890     }
```

tidal_motion

```
1       void tidal_motion(){
2
3         time_i = (60*60*hour()+(60*minute()+second()));
4         time_min_quarter = int(hour()*60*4+minute()*4+floor(second()/15));
5         tide_difference = 0.750;
6         tide_stop_difference = 0.5;
7
8
9         if((time_i>(l_tide_time_today_tm-l_tide_dur_today_tm/2) &&
        time_i<(l_tide_time_today_tm+l_tide_dur_today_tm/2))
10              || (time_i>(h_tide_time_today_tm-
        h_tide_dur_today_tm/2) && time_i<(h_tide_time_today_tm+h_tide_dur_today_tm/
        2))){
11          No_Tide();
12        } else if(time_i>l_tide_time_today_tm && time_i>h_tide_time_today_tm){
13          if(l_tide_time_today_tm>h_tide_time_today_tm){
14            Rising_Tide();
15          }else{
16            Falling_Tide();
17          }
18        } else if(time_i<l_tide_time_today_tm && time_i<h_tide_time_today_tm){
19          if(l_tide_time_today_tm>h_tide_time_today_tm){
20            Rising_Tide();
21          }else{
22            Falling_Tide();
23          }
24        } else if ((time_i>(l_tide_time_today_tm-l_tide_dur_today_tm/2) &&
        time_i<(h_tide_time_today_tm+h_tide_dur_today_tm/2))
25              || (time_i>(h_tide_time_today_tm-
        h_tide_dur_today_tm/2) && time_i<(l_tide_time_today_tm+l_tide_dur_today_tm/
        2))){
26          if(l_tide_time_today_tm>h_tide_time_today_tm){
27            Falling_Tide();
28          }else{
29            Rising_Tide();
30          }
31        } else if ((time_i==(l_tide_time_today_tm-l_tide_dur_today_tm/2)) ||
        (time_i==(l_tide_time_today_tm+l_tide_dur_today_tm/2)) ||
        time_i==(h_tide_time_today_tm-h_tide_dur_today_tm/2) ||
        time_i==(h_tide_time_today_tm+h_tide_dur_today_tm/2)){
32           Stop_Tide();
33        }
34
35      }
36
```

```
37        void Rising_Tide() {
38          //go through mesocosms (1-12)
39          if(poll_i_o==0 && meso_number_n<13){
40            //go through mesocosms that are automated
41            if(Position.getInt((meso_number_n-1),2)==0 &&
      Tide_Records.getFloat(int(hour()*60*4+minute()*4+floor(second()/15)),int(me
      so_number_n-1+17))>-99999){
42              //check what to do for readings
43
44              count=0;
45              marsh_sum=0;
46              for(int ii=0; ii<3;ii++){
47                if(Tide_Records.getFloat(int(hour()*60*4+minute()*4+floor(second(
      )/15)-ii),int(meso_number_n-1+17))!=0 &&
      Tide_Records.getFloat(int(hour()*60*4+minute()*4+floor(second()/15)-
      ii),int(meso_number_n-
      1+17))==Tide_Records.getFloat(int(hour()*60*4+minute()*4+floor(second()/15)
      -ii),int(meso_number_n-1+17))){
48                  marsh_sum = Tide_Records.getFloat(int(hour()*60*4+minute()*4+fl
      oor(second()/15)-ii),int(meso_number_n-1+17))+marsh_sum;
49                  count=count+1;
50                  }
51                }
52              meso_reading_now=marsh_sum/count;
53
54              println("reading_now_r"+meso_reading_now);
55              reading_m = ((4000+R1[meso_number_n-1]-(R1[meso_number_n-
      1]*1023/meso_reading_now))/R2[meso_number_n-1])-offset[meso_number_n-1];
56              println("reading_now_cm"+reading_m);
57              tide_pattern_now = Tide_Records.getFloat(int(time_min_quarter),3);
58              println("slope_reading"+tide_pattern_now);
59              print("diff"+meso_number_n+"-R");
60              println(reading_m-tide_pattern_now);
61              if(tide_pattern_now-reading_m>tide_difference){
62                println(tide_pattern_now-reading_m);
63                println("large");
64                if(reading_m<0){
65                  xbee.write("<"+meso_number_n+","+arduino_signal_rev[meso_number
      _n-1][0]+",>");
66                  println("<"+meso_number_n+","+arduino_signal_rev[meso_number_n-
      1][0]+",>");
67                }else{
68                  xbee.write("<"+meso_number_n+","+arduino_signal_rev_longer[meso
      _number_n-1][0]+",>");
69                  println("<"+meso_number_n+","+arduino_signal_rev_longer[meso_nu
      mber_n-1][0]+",>");
70                }
71                Position.setInt(meso_number_n-1,1,0);
```

```
72                saveTable(Position, "data/PositionData.csv");
73                poll_i_o=1;
74              } else{
75                meso_number_n=meso_number_n+1;
76              }
77            } else{
78              meso_number_n=meso_number_n+1;
79            }
80          }
81        }
82
83        void Falling_Tide() {
84          //go through mesocosms (1-12)
85          if(poll_i_o==0 && meso_number_n<13){
86            //go through mesocosms that are automated
87            if(Position.getInt((meso_number_n-1),2)==0 &&
      Tide_Records.getFloat(int(hour()*60*4+minute()*4+floor(second()/15)),int(me
      so_number_n-1+17))>-99999){
88              //check what to do for readings
89
90              count=0;
91              marsh_sum=0;
92              for(int ii=0; ii<3;ii++){
93                if(Tide_Records.getFloat(int(hour()*60*4+minute()*4+floor(second(
      )/15)-ii),int(meso_number_n-1+17))!=0 &&
      Tide_Records.getFloat(int(hour()*60*4+minute()*4+floor(second()/15)-
      ii),int(meso_number_n-
      1+17))==Tide_Records.getFloat(int(hour()*60*4+minute()*4+floor(second()/15)
      -ii),int(meso_number_n-1+17))){
94                  marsh_sum = Tide_Records.getFloat(int(hour()*60*4+minute()*4+fl
      oor(second()/15)-ii),int(meso_number_n-1+17))+marsh_sum;
95                  count=count+1;
96                  }
97                }
98              meso_reading_now=marsh_sum/count;
99
100             //float meso_reading_now =
      Tide_Records.getFloat(int(hour()*60*4+minute()*4+floor(second()/15)),int(me
      so_number_n-1+17));
101             println("reading_now_f"+meso_reading_now);
102             float reading_m = ((4000+R1[meso_number_n-1]-(R1[meso_number_n-
      1]*1023/meso_reading_now))/R2[meso_number_n-1])-offset[meso_number_n-1];
103             println("reading_now_cm"+reading_m);
104             float tide_pattern_now =
      Tide_Records.getFloat(int(time_min_quarter),3);
105             println("slope_reading"+tide_pattern_now);
106             print("diff"+meso_number_n+"-F");
```

```
107             println(reading_m-tide_pattern_now);
108             if(reading_m-tide_pattern_now>tide_difference){
109               println(tide_pattern_now-reading_m);
110               println("large");
111               if(reading_m<0){
112                 xbee.write("<"+meso_number_n+","+arduino_signal_rev[meso_number
    _n-1][1]+",>");
113               }else{
114                 xbee.write("<"+meso_number_n+","+arduino_signal_rev_longer[meso
    _number_n-1][1]+",>");
115               }
116               Position.setInt(meso_number_n-1,1,1);
117               saveTable(Position, "data/PositionData.csv");
118               poll_i_o=1;
119             }else{
120               meso_number_n=meso_number_n+1;
121             }
122           } else{
123             meso_number_n=meso_number_n+1;
124           }
125         }
126       }
127
128       void No_Tide() {
129          //go through mesocosms (1-12)
130         if(poll_i_o==0 && meso_number_n<13){
131           //go through mesocosms that are automated
132           if(Position.getInt((meso_number_n-1),2)==0 &&
    Tide_Records.getFloat(int(hour()*60*4+minute()*4+floor(second()/15)),int(me
    so_number_n-1+17))>-99999){
133             println("no tides");
134             //check what to do for readings
135             //float meso_reading_now =
    Tide_Records.getFloat(int(hour()*60*4+minute()*4+floor(second()/15)),int(me
    so_number_n-1+17));
136
137
138             count=0;
139             marsh_sum=0;
140             for(int ii=0; ii<3;ii++){
141               if(Tide_Records.getFloat(int(hour()*60*4+minute()*4+floor(second(
    )/15)-ii),int(meso_number_n-1+17))!=0 &&
    Tide_Records.getFloat(int(hour()*60*4+minute()*4+floor(second()/15)-
    ii),int(meso_number_n-
    1+17))==Tide_Records.getFloat(int(hour()*60*4+minute()*4+floor(second()/15)
    -ii),int(meso_number_n-1+17))){
```

```
142                 marsh_sum = Tide_Records.getFloat(int(hour()*60*4+minute()*4+fl
    oor(second()/15)-ii),int(meso_number_n-1+17))+marsh_sum;
143                 count=count+1;
144                 }
145               }
146             meso_reading_now=marsh_sum/count;
147             println("reading_now_n"+meso_reading_now);
148
149             float reading_m = ((4000+R1[meso_number_n-1]-(R1[meso_number_n-
    1]*1023/meso_reading_now))/R2[meso_number_n-1])-offset[meso_number_n-1];
150             println("reading_now_cm"+reading_m);
151             float tide_pattern_now =
    Tide_Records.getFloat(int(time_min_quarter),3);
152             println("slope_reading"+tide_pattern_now);
153             print("diff"+meso_number_n+"-N");
154             println(reading_m-tide_pattern_now);
155             if(tide_pattern_now-reading_m>tide_difference){
156               println(tide_pattern_now-reading_m);
157               println("large");
158               if(reading_m<0){
159                 xbee.write("<"+meso_number_n+","+arduino_signal_rev[meso_number
    _n-1][0]+",>");
160               }else{
161                 xbee.write("<"+meso_number_n+","+arduino_signal_rev_longer[meso
    _number_n-1][0]+",>");
162               }
163               Position.setInt(meso_number_n-1,1,0);
164               saveTable(Position, "data/PositionData.csv");
165               poll_i_o=1;
166             } else if(reading_m-tide_pattern_now>tide_difference){
167               println(tide_pattern_now-reading_m);
168               println("large");
169               if(reading_m<0){
170                 xbee.write("<"+meso_number_n+","+arduino_signal_rev[meso_number
    _n-1][1]+",>");
171               }else{
172                 xbee.write("<"+meso_number_n+","+arduino_signal_rev_longer[meso
    _number_n-1][1]+",>");
173               }
174               Position.setInt(meso_number_n-1,1,1);
175               saveTable(Position, "data/PositionData.csv");
176               poll_i_o=1;
177             }else{
178               meso_number_n=meso_number_n+1;
179             }
180           } else{
```

```
181             meso_number_n=meso_number_n+1;
182           }
183         }
184       }
185       void Stop_Tide() {
186         if(poll_i_o==0 && meso_number_n<13){
187           if(mesocosm_tides_moved_check[meso_number_n-1]==0){
188             xbee.write("<"+meso_number_n+","+arduino_signal[meso_number_n-
    1][2]+",>");
189             println("Stopped Tide for "+meso_number_n);
190             Position.setInt(meso_number_n-1,1,2);
191             saveTable(Position, "data/PositionData.csv");
192             poll_i_o=1;
193             mesocosm_tides_moved_check[meso_number_n-1]=0;
194           }
195         }
196       }
```

record_heights

```
1       void record_heights(){
2         //after we get a reading from the arduinos
3
4         meso_reading_list = split(myStrings, &apos;,&apos;);
5
6         Tide_Records = loadTable("Mesocosm_data_for_"+nf(month(),2)+"_"+nf(d
        ay(),2)+"_"+nf(year(),4)+".csv", "header, csv");
7
8       //  print("Recording ");
9         for(int i=0; i<int(meso_reading_list.length-1);i++){
10
11            meso_reading_list[i]=nf(float(meso_reading_list[i])/15);
12
13      //    if(meso_reading_list[i].equals("")){
14      //      meso_reading_list[i]="0";
15      //   }
16
17          if(reading_m_s==2){
18            //surge tank
19            //check for NaN
20            if(float(meso_reading_list[i])==float(meso_reading_list[i])){
21              Tide_Records.setFloat(int(hour()*60*4+minute()*4+floor(second(
        )/15)),int(i+5),float(meso_reading_list[i]));
22            }else{
23              Tide_Records.setFloat(int(hour()*60*4+minute()*4+floor(second(
        )/15)),int(i+5),0);
24            }
25      //      println("S");
26      //      print(meso_reading_list[i]);
27          }else{
28            //marsh tank
29            //check for NaN
30            if(float(meso_reading_list[i])==float(meso_reading_list[i])){
```

```
31              Tide_Records.setFloat(int(hour()*60*4+minute()*4+floor(second(
    )/15)),int(i+17),float(meso_reading_list[i]));
32          }else{
33              Tide_Records.setFloat(int(hour()*60*4+minute()*4+floor(second(
    )/15)),int(i+17),0);
34          }
35  //        println(meso_reading_list[i]);
36          marsh_h_reading[i]=((4000+R1[i]-
    (R1[i]*1023/float(meso_reading_list[i])))/R2[i])-offset[i];
37  //        println(marsh_h_reading[i]);
38        }
39      }
40  //  println("end rec");
41      saveTable(Tide_Records,
    "data/Mesocosm_data_for_"+nf(month(),2)+"_"+nf(day(),2)+"_"+nf(year(),
    4)+".csv");
42
43      new_data = 1;
44      poll_i_o=0;
45  //  meso_number_n=1;
46      meso_number_i_tide=0;
47
48      //initial water height check
61  }
```

meso_tide_set_page

```
1   void meso_tide_set_page(){
2     fill(200);
3     rect(0,0,width, height);
4     fill(fontcolor);
5     textFont(font, fontsize*2);
6     textAlign(CENTER,CENTER);
7
      text("Set Tides for Mesocosms
    "+Tide_Amp_Data.getString(date_used,0).substring(0,
    Tide_Amp_Data.getString(date_used,0).length())
8     ,width/2, Graph_Y_TL/2);
9
10    //Graph Background
11    fill(250);
12    rect(Graph_X_TL,Graph_Y_TL,Graph_X_Dist,Graph_Y_Dist);
13
14
15
16    //Display buttonto return to main page
17    fill(ColorSetBoxFill);
18    rectMode(CORNER);
19    rect(W*7/8, row, XButton_Main_Page, YButton_Main_Page,5);  //
    Black rectangle
20    fill(SetBoxColor);
21    textAlign(CENTER,CENTER);
22    textFont(font, fontsize*3/4);
23    text("Main Page", W*7/8+XButton_Main_Page/2,
    row+YButton_Main_Page/2);
24
25    //Display Text to Set Tidal Information
26    textFont(font, fontsize);
27    fill(0);
28    textAlign(CENTER,CENTER);
29    for(int i=0;i<Text_Box_String.length;i++){
30      text(Text_Box_String[i],Text_Box_X+col*i,Text_Box_Y);
31    }
32    text("Tide 1",Text_Box_X,Text_Box_Y+row);
```

```
33        text("Tide 2",Text_Box_X,Text_Box_Y+row*2);
34
          text("Click to Set Tidal Information.  Enter to confirm. \n
        Backspace to cancel.  24 Hour
        Time",Text_Box_X+col*1.5,Text_Box_Y+row*3);
35
36        //Display Buttons to Set Tidal Information
37        for(int i = 1;i<4;i++){
38          for(int ii = 1;ii<3;ii++){
39            fill(250);
40
              rect(Text_Box_X+col*i-XButton_Main_Page/2,Text_Box_Y+row*ii-
        YButton_Main_Page/2,XButton_Main_Page,YButton_Main_Page);
41            fill(0);
42
              text(nf(Tide_Amp_Data.getFloat(date_used,i+((ii-
        1)*3)),2,1),Text_Box_X+col*i,Text_Box_Y+row*ii);
43            if(Tide_Amp_Data.getFloat(today_date_int,1)<Tide_Amp_Data.ge
        tFloat(today_date_int,4)){
44              graph_variables[(i-1)+(ii-
        1)*3]=Tide_Amp_Data.getFloat(date_used,i+((ii-1)*3));
45            } else {
46              graph_variables[(i-1)+(ii-2)*-
        3]=Tide_Amp_Data.getFloat(date_used,i+((ii-1)*3));
47            }
48          }
49        }
50
51        //Display Buttons
52
53        fill(250);
54
          rect(Text_Box_X+col*4-XButton_Main_Page/2,Text_Box_Y+row-
        YButton_Main_Page/2,XButton_Main_Page,YButton_Main_Page,5);
55        fill(0);
56        text("Set",Text_Box_X+col*4,Text_Box_Y+row);
57
58        fill(250);
```

```
59
      rect(Text_Box_X+col*4-XButton_Main_Page/2,Text_Box_Y+row*2-
    YButton_Main_Page/2,XButton_Main_Page,YButton_Main_Page,5);
60      fill(0);
61      text("Set Week",Text_Box_X+col*4,Text_Box_Y+row*2);
62
63      fill(250);
64      rect(Text_Box_X+col*4-XButton_Main_Page/2,Text_Box_Y+row*3-
    YButton_Main_Page/2,XButton_Main_Page,YButton_Main_Page,5);
65      fill(0);
66      text("Set Month",Text_Box_X+col*4,Text_Box_Y+row*3);
67
68      fill(250);
69      rect(Text_Box_X+col*5-XButton_Main_Page/2,Text_Box_Y+row-
    YButton_Main_Page/2,XButton_Main_Page,YButton_Main_Page,5);
70      fill(0);
71      text("Prev. Day",Text_Box_X+col*5,Text_Box_Y+row);
72
73      fill(250);
74      rect(Text_Box_X+col*5-XButton_Main_Page/2,Text_Box_Y+row*2-
    YButton_Main_Page/2,XButton_Main_Page,YButton_Main_Page,5);
75      fill(0);
76      text("Next Day",Text_Box_X+col*5,Text_Box_Y+row*2);
77
78      //Display Lines for Tide of the Day
79      println(graph_variables);
80
    //  float map_x0 = Graph_X_TL+map(graph_variables[0]-
    graph_variables[2]/2,24,48,0,Graph_X_Dist);
81  //  float map_y0 = Graph_Y_TL+map(30-
    graph_variables[4],0,60,0,Graph_Y_Dist);
82
83
84
```

```
85
      float map_x0 = Graph_X_TL+map(2*graph_variables[0]-
    graph_variables[3]+graph_variables[5]/2,0,24,0,Graph_X_Dist);
86    float map_y0 = Graph_Y_TL+map(30-
    graph_variables[4],0,60,0,Graph_Y_Dist);
87    float map_x1 = Graph_X_TL+map(graph_variables[0]-
    graph_variables[2]/2,0,24,0,Graph_X_Dist);
88    float map_y1 = Graph_Y_TL+map(30-
    graph_variables[1],0,60,0,Graph_Y_Dist);
89    float map_x2 =
    Graph_X_TL+map(graph_variables[0]+graph_variables[2]/2,0,24,0,Grap
    h_X_Dist);
90    float map_y2 = Graph_Y_TL+map(30-
    graph_variables[1],0,60,0,Graph_Y_Dist);
91    float map_x3 = Graph_X_TL+map(graph_variables[3]-
    graph_variables[5]/2,0,24,0,Graph_X_Dist);
92    float map_y3 = Graph_Y_TL+map(30-
    graph_variables[4],0,60,0,Graph_Y_Dist);
93    float map_x4 =
    Graph_X_TL+map(graph_variables[3]+graph_variables[5]/2,0,24,0,Grap
    h_X_Dist);
94    float map_y4 = Graph_Y_TL+map(30-
    graph_variables[4],0,60,0,Graph_Y_Dist);
95    float map_x5 =
    Graph_X_TL+map(2*graph_variables[3]+graph_variables[5]/2-
    graph_variables[0]-graph_variables[2]/2,0,24,0,Graph_X_Dist);
96    float map_y5 = Graph_Y_TL+map(30-
    graph_variables[1],0,60,0,Graph_Y_Dist);
97
98    stroke(250,0,0);
99    strokeWeight(4);
100   line(map_x0,map_y0,map_x1,map_y1);
101   line(map_x1,map_y1,map_x2,map_y2);
102   line(map_x2,map_y2,map_x3,map_y3);
103   line(map_x3,map_y3,map_x4,map_y4);
104   line(map_x4,map_y4,map_x5,map_y5);
105   stroke(0);
106   strokeWeight(1);
107   textAlign(LEFT,BOTTOM);
108   fill(0);
109   text("Updated
    "+nf(hour(),2)+":"+nf(minute()+1,2),9*width/11,height);
110
```

```
111     //Grey out sides of graph
112     noStroke();
113     fill(200);
114     rect(0,0,Graph_X_TL,Graph_Y_TL+Graph_Y_Dist);
115     fill(200);
116     rect(Graph_X_TL+Graph_X_Dist,Graph_Y_TL,Graph_X_TL,Graph_Y_TL+Gr
        aph_Y_Dist);
117     stroke(1);
118
119     //X Axis
120     for(int i=0; i<8;i++){
121       fill(0);
122       rect(Graph_X_Side_Bar-X_Axis_X,
        Graph_Y_TL+Graph_Y_Dist*i/8,X_Axis_X,X_Axis_Y);
123       textFont(font, fontsize);
124       textAlign(CENTER, CENTER);
125       text(Depths[i],Graph_X_Side_Bar/2,
        Graph_Y_TL+Graph_Y_Dist*i/8);
126       if(i==4){
127         strokeWeight(3);
128       }if(i!=4){
129         strokeWeight(1);
130       }
131         line(Graph_X_TL,Graph_Y_TL+2+Graph_Y_Dist*i/8,Graph_X_BR,Gra
        ph_Y_TL+2+Graph_Y_Dist*i/8);
132     }
133     fill(0);
134     rect(Graph_X_Side_Bar-X_Axis_X,Graph_Y_BR-
        X_Axis_Y,X_Axis_X,X_Axis_Y);
135     textFont(font, fontsize);
136     text(Depths[8],Graph_X_Side_Bar/2,Graph_Y_BR-X_Axis_Y);
137
138     //Y Axis
139     for(int i=0; i<4;i++){
140       fill(0);
141       rect(Graph_X_Side_Bar+i*(Graph_X_Dist)/4,Graph_Y_BR,Y_Axis_X,Y
        _Axis_Y);
142       textFont(font, fontsize);
143       text(Times[i],Graph_X_Side_Bar+2+i*(Graph_X_Dist)/4,Graph_Y_BR
        +35);
144     }
```

```
145     for(int i=1; i<4;i++){
146       line(Graph_X_Side_Bar+3+i*(Graph_X_Dist)/4,Graph_Y_TL,Graph_X_
    Side_Bar+3+i*(Graph_X_Dist)/4,Graph_Y_BR);
147     }
148     rect(Graph_X_BR-Y_Axis_X,Graph_Y_BR,Y_Axis_X,Y_Axis_Y);
149     textFont(font, fontsize);
150     text(Times[4],Graph_X_BR-Y_Axis_X,Graph_Y_BR+35);
151
152   }
```

Manual_Override

```
1   void manual_override(){
2
3     fill(200);
4     rect(0,0,width, height);
5     fill(fontcolor);
6
7   background(200);
8
9     pushMatrix();
10    translate(width/2,height/6);
11    scale(.25);
12    image(Mesocosms,0,0);
13    popMatrix();
14  int Meso=1;
15
16    fill(ColorSetBoxFill);
17    rectMode(CORNER);
18    rect(W*7/8, row, XButton_Main_Page, YButton_Main_Page,5);  // Black
    rectangle
19    fill(SetBoxColor);
20    textAlign(CENTER,CENTER);
21    textFont(font, fontsize*3/4);
22    text("Main Menu", W*7/8+XButton_Main_Page/2,
    row+YButton_Main_Page/2);
23
24  for(int iii=1; iii<13;iii++){
25
26    pushMatrix();
27    translate(XRow1, YRow1+Yspace*(Meso-1));
28    //draw font
29    fill(fontcolor);
30    textFont(font, fontsize);
31    textAlign(CENTER,CENTER);
32    text("Mesocosm "+Meso, XrefText, YrefText);
33
34    pushMatrix();
35    //draw border
36    translate(XASButton+XASButton/4, 0);
37    if (MesoRow[Meso-1]==1){
38      fill(colorASButton/2);
39      }else{
```

```
40          fill(colorASButton);
41          }
42      rect(0, 0, XASButton, YASButton,5);  // Black rectangle
43      //draw font
44      fill(fontcolor);
45      textFont(font, fontsize);
46      textAlign(CENTER,CENTER);
47      text(tide_status_selection[0], XrefText, YrefText);
48      pushMatrix();
49      translate(XASButton+XASButton/4, 0);
50      if (MesoRow[Meso-1]==2){
51          fill(colorASButton/2);
52          }else{
53          fill(colorASButton);
54          }
55      rect(0, 0, XASButton, YASButton,5);
56      //draw font
57      fill(fontcolor);
58      textFont(font, fontsize);
59      textAlign(CENTER,CENTER);
60      text(tide_status_selection[1], XrefText, YrefText);
61
62      pushMatrix();
63      translate(XASButton+XASButton/4, 0);
64      if (MesoRow[Meso-1]==3){
65          fill(colorASButton/2);
66          }else{
67          fill(colorASButton);
68          }
69      rect(0, 0, XASButton, YASButton,5);  // Black rectangle
70      //draw font
71      fill(fontcolor);
72      textFont(font, fontsize);
73      textAlign(CENTER,CENTER);
74      text(tide_status_selection[2], XrefText, YrefText);
75
76      pushMatrix();
77      //manual set button
78      translate(XASButton+XASButton/4, 0);
```

```
79      fill(150*(Position.getInt(Meso-1,2)),150,150*(Position.getInt(Meso-
      1,2)));
80      rect(0, 0, XASButton, YASButton,5);  // Black rectangle
81      //draw font
82      fill(fontcolor);
83      textFont(font, fontsize);
84      textAlign(CENTER,CENTER);
85      text(tide_auto_manual[Position.getInt(Meso-1,2)], XrefText,
      YrefText);
86
87      pushMatrix();
88      translate(XASButton+XASButton/4, 0);
89      if (MesoRow[Meso-1]==5){
90        fill(colorASButton/2);
91        MesoRow[Meso-1]=0;
92        }else{
93        fill(colorASButton);
94        }
95      rect(0, 0, XSButton, YSButton,5);  // Black rectangle
96      //draw font
97      fill(fontcolor);
98      textFont(font, fontsize);
99      textAlign(CENTER,CENTER);
100     text("Set", XSrefText, YSrefText);
101
102     //This displays what option is being pressed
103     pushMatrix();
104     translate(XSButton+XASButton/4, 0);
105       fill(colorSButton);
106     rect(0, 0, XASButton, YASButton);  // Black rectangle
107     //draw font
108     fill(fontcolor);
109     textFont(font, fontsize);
110     textAlign(CENTER,CENTER);
111     text(tide_status_selection[Position.getInt(Meso-1,1)], XrefText,
      YrefText);
112
113     popMatrix();
114     popMatrix();
115     popMatrix();
116     popMatrix();
117     popMatrix();
```

```
118     popMatrix();
119     popMatrix();
120
121   Meso=Meso+1;
122   }
123
124   }
```

main_page

```
1       void main_page(){
2
3
4         fill(200);
5         rect(0,0,width, height);
6
7         //define font
8         fill(fontcolor);
9         textFont(font, fontsize);
10        textAlign(CENTER,CENTER);
11
12
13        //Display mesocosm numbering image.
14        pushMatrix();
15          translate(width/2,height/7+10);
16          scale(.25);
17          image(Mesocosms,0,0);
18        popMatrix();
19
20        //Display buttons for setting tides
21        fill(ColorSetBoxFill);
22        rectMode(CORNER);
23        rect(W*7/8, row, XButton_Main_Page, YButton_Main_Page,5);  // Black
        rectangle
24        fill(SetBoxColor);
25        textAlign(CENTER,CENTER);
26        textFont(font, fontsize*3/4);
27        text("Set Tides", W*7/8+XButton_Main_Page/2,
        row+YButton_Main_Page/2);
28          //manual set tides
29        fill(ColorSetBoxFill);
30        rectMode(CORNER);
31        rect(W*7/8, row*2, XButton_Main_Page, YButton_Main_Page,5);  //
        Black rectangle
32        fill(SetBoxColor);
33        textAlign(CENTER,CENTER);
34        textFont(font, fontsize*3/4);
35        text("Override", W*7/8+XButton_Main_Page/2,
        row*2+YButton_Main_Page/2);
36
37        //Display mesocosm tables
38        pushMatrix();
39          textAlign(CENTER,CENTER);
40          translate(width/5,height/3);
41          //display header
42          textFont(font, fontsize*1.1);
43          for(int i=0;i<5;i++){
44            text(header[i],(col+13)*i,0);
45          }
46          textFont(font, fontsize*3/4);
47
48          //display list of mesocosms (1-12)
49          for(int i=1;i<13;i++){
50            textFont(font, fontsize);
```

```
51            textAlign(CENTER,CENTER);
52            text("Mesocosm "+i, 0, row*i);
53
54            //display buttons
55            //Create Set Box for Lowering Tide
56            fill(ColorSetBoxFill);
57            rectMode(CORNER);
58            rect(col*4, row*i, XButton_Main_Page, YButton_Main_Page,5);  //
      Black rectangle
59            fill(SetBoxColor);
60            textAlign(CENTER,CENTER);
61            textFont(font, fontsize*3/4);
62            text("Meso. Page", col*4+XButton_Main_Page/2,
      row*i+YButton_Main_Page/2);
63
64          }
65    //    popMatrix();
66    //    pushMatrix();
67          translate(XASButton+XASButton/4+col*2, 0);
68          for(int i=1;i<13;i++){
69            //manual or auto set icon
70            translate(0, row);
71            fill(150*(Position.getInt(i-1,2)),150,150*(Position.getInt(i-
      1,2)));
72            rect(0, 0, XASButton, YASButton);  // Black rectangle
73            //draw font
74            fill(fontcolor);
75            textFont(font, fontsize);
76            textAlign(CENTER,CENTER);
77            text(tide_auto_manual[Position.getInt(i-1,2)], XrefText,
      YrefText);
78          }
79
80        popMatrix();
81
82        textAlign(LEFT,BOTTOM);
83        text("Updated
      "+nf(hour(),2)+":"+nf(minute()+1,2),9*width/11,height);
84
85    }
```

individual_meso_page

```
1       void individual_meso_page(){
2         fill(200);
3         rect(0,0,width, height);
4         fill(fontcolor);
5         textFont(font, fontsize*2);
6         textAlign(CENTER,CENTER);
7         text("Mesocosm "+meso_Page,width/2, Arrow_LR_Meso_Y_Loc_1_mp);
8         imageMode(CENTER);
9         text("for",width/2,Arrow_LR_Meso_Y_Loc_1_mp+Arrow_LR_Y_Dist);
10
11        pushMatrix();
12          translate(Arrow_LR_Meso_X_Loc_1_mp,Arrow_LR_Meso_Y_Loc_1_mp+1);
13          image(Arrow_LR,0, 0,Arrow_LR_X_Dist,Arrow_LR_Y_Dist);
14          pushMatrix();
15            translate(Arrow_LR_Meso_X_Loc_2,0);
16            scale(-1,1);
17            image(Arrow_LR,0, 0,Arrow_LR_X_Dist,Arrow_LR_Y_Dist);
18          popMatrix();
19        popMatrix();
20
21        text(filenames_M[ref_Mesocosm_Data_n],width*1/2,Arrow_LR_Meso_Y_L
      oc_1_mp+Arrow_LR_Y_Dist*2);
22
23        pushMatrix();
24          translate(Arrow_LR_Meso_X_Loc_1_mp,Arrow_LR_Meso_Y_Loc_1_mp+Arr
      ow_LR_Y_Dist*2);
25          image(Arrow_LR,0, 0,Arrow_LR_X_Dist,Arrow_LR_Y_Dist);
26          pushMatrix();
27            translate(Arrow_LR_Meso_X_Loc_2,0);
28            scale(-1,1);
29            image(Arrow_LR,0, 0,Arrow_LR_X_Dist,Arrow_LR_Y_Dist);
30          popMatrix();
31        popMatrix();
32
33        //Graph Background
34        fill(250);
35        rect(Graph_X_TL,Graph_Y_TL_mp,Graph_X_Dist,Graph_Y_Dist_mp);
36
37          //X Axis
38        for(int i=0; i<8;i++){
39          fill(0);
40          rect(Graph_X_Side_Bar-X_Axis_X,
      Graph_Y_TL_mp+Graph_Y_Dist_mp*i/8,X_Axis_X,X_Axis_Y);
41          textFont(font, fontsize);
42          text(Depths[i],Graph_X_Side_Bar/2,
      Graph_Y_TL_mp+Graph_Y_Dist_mp*i/8);
43          if(i==4){
44            strokeWeight(3);
45          }if(i!=4){
46            strokeWeight(1);
```

```
47          }
48            line(Graph_X_TL,Graph_Y_TL_mp+2+Graph_Y_Dist_mp*i/8,Graph_X_B
      R,Graph_Y_TL_mp+2+Graph_Y_Dist_mp*i/8);
49        }
50        fill(0);
51        rect(Graph_X_Side_Bar-X_Axis_X,Graph_Y_BR_mp-
      X_Axis_Y,X_Axis_X,X_Axis_Y);
52        textFont(font, fontsize);
53        text(Depths[8],Graph_X_Side_Bar/2,Graph_Y_BR_mp-X_Axis_Y);
54
55        //Y Axis
56        for(int i=0; i<4;i++){
57          fill(0);
58          rect(Graph_X_Side_Bar+i*(Graph_X_Dist)/4,Graph_Y_BR_mp,Y_Axis_X
      ,Y_Axis_Y);
59            textFont(font, fontsize);
60            text(Times[i],Graph_X_Side_Bar+2+i*(Graph_X_Dist)/4,Graph_Y_BR_
      mp+35);
61        }
62        for(int i=1; i<4;i++){
63        line(Graph_X_Side_Bar+3+i*(Graph_X_Dist)/4,Graph_Y_TL_mp,Graph_X_
      Side_Bar+3+i*(Graph_X_Dist)/4,Graph_Y_BR_mp);
64        }
65        rect(Graph_X_BR-Y_Axis_X,Graph_Y_BR_mp,Y_Axis_X,Y_Axis_Y);
66        textFont(font, fontsize);
67        text(Times[4],Graph_X_BR-Y_Axis_X,Graph_Y_BR_mp+35);
68
69        noStroke();
70
71        for(int i=0;i<1440;i=i+1){
72          if(Tide_Records.getFloat(i*4,3)>-999999){
73            fill(250,0,0);
74            float y_int = Tide_Records.getFloat(i*4,3);
75            ellipse(map(i,0,1440,Graph_X_TL,Graph_X_TL+Graph_X_Dist),map(
      y_int,30,-30,Graph_Y_TL_mp,Graph_Y_TL_mp+Graph_Y_Dist_mp),5,5);
76          }
77          if(Tide_Records.getFloat(i*4,meso_Page+16)>-999999){
78            fill(0,0,250);
79            //average of three previous points
80            count=0;
81            y_marsh_sum=0;
82            for(int ii=0; ii<3;ii++){
83              if(Tide_Records.getFloat(i*4-ii,meso_Page+16)!=0 &&
      Tide_Records.getFloat(i*4-
      ii,meso_Page+16)==Tide_Records.getFloat(i*4-ii,meso_Page+16)){
84                y_marsh_sum = Tide_Records.getFloat(i*4-
      ii,meso_Page+16)+y_marsh_sum;
85                count=count+1;
86              }
87              y_marsh=y_marsh_sum/count;
88            }
89
```

```
90          ellipse(map(i,0,1440,Graph_X_TL,Graph_X_TL+Graph_X_Dist),map(
      ((4000+R1[meso_Page-1]-(R1[meso_Page-
      1]*1023/y_marsh))/R2[meso_Page-1])-offset[meso_Page-1],30,-
      30,Graph_Y_TL_mp,Graph_Y_TL_mp+Graph_Y_Dist_mp),5,5);;
91          }
92        }
93        //display mesocosm information
94
95        fill(0);
96        textAlign(LEFT,CENTER);
97        text("Marsh Water Reading",Text_Box_X,ind_meso_information);
98        text(nf(reading_m,2,1)+"
      cm",Graph_X_Side_Bar+1*(Graph_X_Dist)/4,ind_meso_information);
99
100       fill(0);
101       textAlign(LEFT,CENTER);
102       text("Water Height
      Target",Text_Box_X,ind_meso_information+XRow1);
103       text(nf(tide_pattern_now,2,1)+"
      cm",Graph_X_Side_Bar+1*(Graph_X_Dist)/4,ind_meso_information+XRow1)
      ;
104
105        fill(0);
106       textAlign(LEFT,CENTER);
107       text("Status",Text_Box_X,ind_meso_information+XRow1*2);
108       text(tide_auto_manual[Position.getInt(meso_Page-
      1,2)],Graph_X_Side_Bar+1*(Graph_X_Dist)/4,ind_meso_information+XRow
      1*2);
109       text(tide_status_selection[Position.getInt(meso_Page-
      1,1)],Graph_X_Side_Bar+2*(Graph_X_Dist)/4,ind_meso_information+XRow
      1*2);
110
111       stroke(0);
112       //Display buttonto return to main page
113
114       fill(ColorSetBoxFill);
115       rectMode(CORNER);
116       rect(W*7/8, row, XButton_Main_Page, YButton_Main_Page,5);  //
      Black rectangle
117       fill(SetBoxColor);
118       textAlign(CENTER,CENTER);
119       textFont(font, fontsize*3/4);
120       text("Main Menu", W*7/8+XButton_Main_Page/2,
      row+YButton_Main_Page/2);
121
122       fill(ColorSetBoxFill);
123       rectMode(CORNER);
124       rect(W*7/8, row*2, XButton_Main_Page, YButton_Main_Page,5);  //
      Black rectangle
125       fill(SetBoxColor);
126       textAlign(CENTER,CENTER);
127       textFont(font, fontsize*3/4);
128       text("Set Tide", W*7/8+XButton_Main_Page/2,
      row*2+YButton_Main_Page/2);
```

```
129
130       fill(ColorSetBoxFill);
131       rectMode(CORNER);
132       rect(W*7/8, row*3, XButton_Main_Page, YButton_Main_Page,5);  //
      Black rectangle
133       fill(SetBoxColor);
134       textAlign(CENTER,CENTER);
135       textFont(font, fontsize*3/4);
136       text("Override", W*7/8+XButton_Main_Page/2,
      row*3+YButton_Main_Page/2);
137
138       /*
139       fill(ColorSetBoxFill);
140       rectMode(CORNER);
141       rect(W*2/3, YButton_Main_Page, XButton_Main_Page, YButton_Main_Pa
      ge);  // Black rectangle
142       fill(SetBoxColor);
143       textAlign(CENTER,CENTER);
144       textFont(font, fontsize*3/4);
145       text("Main Page", W*2/3+XButton_Main_Page/2, YButton_Main_Page);
146       */
147       textAlign(LEFT,BOTTOM);
148       text("Updated
      "+nf(hour(),2)+":"+nf(minute()+1,2),9*width/11,height);
149
150     }
```

file_check

```
1       void file_check(){
2         int create=0;
3         int create_tide_amp =0;
4         //This checks if a new file needs to be created for positions to
        be saved, creates one if needed or loads table if not
5
6         // Check for table; list names here
7         String path = sketchPath("data");
8         println("Listing all filenames in a directory: ");
9         println(sketchPath("data"));
10        String[] filenames = listFileNames(path);
11        printArray(filenames);
12
13        //goes through and checks for table name
14
15        for (int i=0; i<filenames.length;i++){
16          if(filenames[i].equals("PositionData.csv")){
17          create=1;
18          }
19        }
20
21        for (int i=0; i<filenames.length;i++){
22          if(filenames[i].equals("Tide_Amp_Data.csv")){
23          create_tide_amp=1;
24          }
25        }
26
27        //add date check here
28
29        if(create_tide_amp==0){
30          int [] days_month = {31,28,31,30,31,30,31,31,30,31,30,31};
31          cursor(WAIT);
32          Tide_Amp_Data = new Table();
33          Tide_Amp_Data.addColumn("Date");
34          Tide_Amp_Data.addColumn("Tide 1 Duration");
35          Tide_Amp_Data.addColumn("Tide 1 Amp. Tide");
36          Tide_Amp_Data.addColumn("Tide 1 Height");
37          Tide_Amp_Data.addColumn("Tide 2 Duration");
38          Tide_Amp_Data.addColumn("Tide 2 Amp. Tide");
39          Tide_Amp_Data.addColumn("Tide 2 Height");
40          int date = 0;
41          //create rows for dates
42          if(year()%4==0){
43            days_month[2]=int(days_month[2]+1);
44          }
45          for(int i=0;i<12;i++){
46            for(int ii=1;ii<=days_month[i];ii++){
47              date=date+1;
48              Tide_Amp_Data.addRow();
```

```
49              Tide_Amp_Data.setString(date,0, int(i+1)+"-"+ii);
50              for(int iii=1;iii<7;iii++){
51                Tide_Amp_Data.setFloat(date,iii, 0);
52              }
53              if(month()==i && day()==ii){
54                today_date_int = date;
55                date_used = today_date_int;
56              }
57            }
58          }
59          saveTable(Tide_Amp_Data, "data/Tide_Amp_Data.csv");
60
61        }
62
63        if(create_tide_amp==1){
64          Tide_Amp_Data = loadTable("Tide_Amp_Data.csv", "header, csv");
65          println("loaded");
66          saveTable(Tide_Amp_Data, "data/Tide_Amp_Data.csv");
67
68          int date = 0;
69          for(int i=0;i<12;i++){
70            int [] days_month = {31,28,31,30,31,30,31,31,30,31,30,31};
71            for(int ii=1;ii<=days_month[i];ii++){
72              date=date+1;
73              Tide_Amp_Data.addRow();
74              Tide_Amp_Data.setString(date,0, int(i+1)+"-"+ii);
75              if(month()==(i+1) && day()==ii){
76                today_date_int = date;
77                date_used = today_date_int;
78              }
79            }
80          }
81        }
82
83        //check for table
84
85        if(create==0){
86
87        // if table is needed, create
88
89        println("Table Created");
90        cursor(WAIT);
91
92        Position = new Table();
93        Position.addColumn("Mesocosm");
94        Position.addColumn("Position");
95        Position.addColumn("Manual_Auto");
96        for(int m=1;m<13;m=m+1){
97        TableRow mesocosm = Position.addRow();
98        mesocosm.setString("Mesocosm","Mesocosm "+m);
99        mesocosm.setInt("Position",0);
100       mesocosm.setInt("Manual_Auto",1);
```

```
101      }
102      saveTable(Position, "data/PositionData.csv");
103
104      //if table is not needed, tell user
105
106      }
107      if(create==1){
108      println("new table not needed");
109      Position = loadTable("PositionData.csv", "header, csv");
110      saveTable(Position, "data/PositionData.csv");
111      }
112
113      //check for this date&apos;s file
114
115      //goes through and checks for table name
116      create=0;
117
118
119      Table Tide_Records;
120
121      for (int i=0; i<filenames.length;i++){
122        if(filenames[i].equals("Mesocosm_data_for_"+Date_Text+"_"+nf(y
       ear(),4)+".csv")){
123        create=1;
124        }
125      }
126
127      if (create==0){
128      println("Table Created");
129      cursor(WAIT);
130      Tide_Records = new Table();
131
132      Tide_Records.addColumn("Time_Set");
133      Tide_Records.addColumn("Time_Point");
134      Tide_Records.addColumn("Amplitude_Set");
135      Tide_Records.addColumn("Meso_Tides");
136      for(int i=1;i<14;i++){
137        if(i==1){
138        Tide_Records.addColumn("Time_of_Day");
139        }else{
140        Tide_Records.addColumn("Surge_"+str(int(i-1))+"_Reading");
141        }
142      }
143      for(int i=1;i<13;i++){
144        Tide_Records.addColumn("Marsh_"+str(int(i))+"_Reading");
145      }
146      /*
147      TableRow Point_1 = Tide_Records.addRow();
148      Point_1.setInt("Time_Set", 0);
149      Point_1.setInt("Time_Point", tides[0]);
150      Point_1.setInt("Amplitude_Set", tidal_amplitude[0]);
151
```

```
152     TableRow Point_2 = Tide_Records.addRow();
153     Point_2.setInt("Time_Set", 0);
154     Point_2.setInt("Time_Point", tides[1]);
155     Point_2.setInt("Amplitude_Set", tidal_amplitude[1]);
156
157     TableRow Point_3 = Tide_Records.addRow();
158     Point_3.setInt("Time_Set", 0);
159     Point_3.setInt("Time_Point", tides[2]);
160     Point_3.setInt("Amplitude_Set", tidal_amplitude[2]);
161
162     TableRow Point_4 = Tide_Records.addRow();
163     Point_4.setInt("Time_Set", 0);
164     Point_4.setInt("Time_Point", tides[3]);
165     Point_4.setInt("Amplitude_Set",tidal_amplitude[3]);
166     */
167       for(int i=0;i<5760;i=i+1){
168         Tide_Records.addRow();
169         Tide_Records.setInt(int(i),4, i);
170       }
171       //get data.  first low, then high
172       //get hours.
173
174         float i_tide_amp_today =
    Tide_Amp_Data.getFloat(today_date_int,2);
175         float ii_tide_amp_today =
    Tide_Amp_Data.getFloat(today_date_int,5);
176
177         if(i_tide_amp_today<=ii_tide_amp_today){
178           l_tide_amp_today = Tide_Amp_Data.getFloat(date_used,2);
179           h_tide_amp_today = Tide_Amp_Data.getFloat(date_used,5);
180           l_tide_time_today = Tide_Amp_Data.getFloat(date_used,1);
181           h_tide_time_today = Tide_Amp_Data.getFloat(date_used,4);
182           l_tide_dur_today = Tide_Amp_Data.getFloat(date_used,3);
183           h_tide_dur_today = Tide_Amp_Data.getFloat(date_used,6);
184         }else{
185           l_tide_amp_today = Tide_Amp_Data.getFloat(date_used,5);
186           h_tide_amp_today = Tide_Amp_Data.getFloat(date_used,2);
187           l_tide_time_today = Tide_Amp_Data.getFloat(date_used,4);
188           h_tide_time_today = Tide_Amp_Data.getFloat(date_used,1);
189           l_tide_dur_today = Tide_Amp_Data.getFloat(date_used,6);
190           h_tide_dur_today = Tide_Amp_Data.getFloat(date_used,3);
191         }
192
193       for(int i=0;i<5760;i=i+1){
194         if(i>4*24*60*(h_tide_time_today-h_tide_dur_today/2) &&
    i<4*24*60*(h_tide_time_today+h_tide_dur_today/2)){
195           Tide_Records.setFloat(i ,3 , h_tide_amp_today);
196         }
197         if(i>4*24*60*(l_tide_time_today-l_tide_dur_today/2) &&
    i<4*24*60*(l_tide_time_today+l_tide_dur_today/2)){
198           Tide_Records.setFloat(i ,3 , l_tide_amp_today);
199         }
```

```
200       }
201
202       saveTable(Tide_Records,
      "data/Mesocosm_data_for_"+Date_Text+"_"+nf(year(),4)+".csv");
203     }
204     if (create==1){
205       println("new table not needed");
206       Tide_Records = loadTable("Mesocosm_data_for_"+nf(month(),2)+"_
      "+nf(day(),2)+"_"+nf(year(),4)+".csv", "header, csv");
207       println("loaded");
208       saveTable(Tide_Records,
      "data/Mesocosm_data_for_"+nf(month(),2)+"_"+nf(day(),2)+"_"+nf(yea
      r(),4)+".csv");
209   //    for(int i=0;i<4;i++){
210   //      tides[i]=Tide_Records.getInt(i,1);
211   //      tidal_amplitude[i]=Tide_Records.getInt(i,2);
212   //    }
213     }
214
215       cursor(ARROW);
216   }
```

file_check_named

```
1     void file_check_named(String date_to_be_checked){
2
3       int create=0;
4
5       String Date_Text = date_to_be_checked;
6
7       println("Banana");
8       println(Date_Text);
9       println("Banana");
10
11      //This checks if a new file needs to be created for positions to
      be saved, creates one if needed or loads table if not
12
13      // Check for table; list names here
14      String path = sketchPath("data");
15      println("Listing all filenames in a directory: ");
16      println(sketchPath("data"));
17      String[] filenames = listFileNames(path);
18      printArray(filenames);
19
20      //goes through and checks for table name
21
22      Table Tide_Records;
23
24      for (int i=0; i<filenames.length;i++){
25        if(filenames[i].equals("Mesocosm_data_for_"+Date_Text+"_"+nf(y
      ear(),4)+".csv")){
26        create=1;
27        println(Date_Text);
28        }
29      }
30
31      if (create==0){
32      println("Table Created");
33      cursor(WAIT);
34      Tide_Records = new Table();
35
36      Tide_Records.addColumn("Time_Set");
37      Tide_Records.addColumn("Time_Point");
38      Tide_Records.addColumn("Amplitude_Set");
39      Tide_Records.addColumn("Meso_Tides");
40      for(int i=1;i<14;i++){
41        if(i==1){
42        Tide_Records.addColumn("Time_of_Day");
43        }else{
44        Tide_Records.addColumn("Surge_"+str(int(i-1))+"_Reading");
45        }
46      }
47      for(int i=1;i<13;i++){
48        Tide_Records.addColumn("Marsh_"+str(int(i))+"_Reading");
```

```
49      }
50
51        for(int i=0;i<5760;i=i+1){
52          Tide_Records.addRow();
53          Tide_Records.setInt(int(i),4, i);
54        }
55        //get data.  first low, then high
56        //get hours.
57
58          int today_used_calcs=today_date_int+1;
59
60          float i_tide_time_yesterday =
      Tide_Amp_Data.getFloat(today_used_calcs-1,1);
61          float ii_tide_time_yesterday =
      Tide_Amp_Data.getFloat(today_used_calcs-1,4);
62          if(i_tide_time_yesterday<=ii_tide_time_yesterday){
63            tide_amp_yesterday = Tide_Amp_Data.getFloat(today_used_cal
      cs-1,5);
64            tide_time_yesterday = Tide_Amp_Data.getFloat(today_used_ca
      lcs-1,4);
65            tide_dur_yesterday = Tide_Amp_Data.getFloat(today_used_cal
      cs-1,6);
66          }else{
67            tide_amp_yesterday = Tide_Amp_Data.getFloat(today_used_cal
      cs-1,2);
68            tide_time_yesterday = Tide_Amp_Data.getFloat(today_used_ca
      lcs-1,1);
69            tide_dur_yesterday = Tide_Amp_Data.getFloat(today_used_cal
      cs-1,3);
70          }
71
72          float tide_end_yesterday =
      tide_time_yesterday+tide_dur_yesterday/2;
73
74          float i_tide_time_tomorrow =
      Tide_Amp_Data.getFloat(today_used_calcs+1,1);
75          float ii_tide_time_tomorrow =
      Tide_Amp_Data.getFloat(today_used_calcs+1,4);
76          if(ii_tide_time_tomorrow<=i_tide_time_tomorrow){
77            tide_amp_tomorrow = Tide_Amp_Data.getFloat(today_used_calc
      s+1,5);
78            tide_time_tomorrow = Tide_Amp_Data.getFloat(today_used_cal
      cs+1,4);
79            tide_dur_tomorrow = Tide_Amp_Data.getFloat(today_used_calc
      s+1,6);
80          }else{
81            tide_amp_tomorrow = Tide_Amp_Data.getFloat(today_used_calc
      s+1,2);
82            tide_time_tomorrow = Tide_Amp_Data.getFloat(today_used_cal
      cs+1,1);
83            tide_dur_tomorrow = Tide_Amp_Data.getFloat(today_used_calc
      s+1,3);
84          }
85
```

```
86          float tide_start_tomorrow = tide_time_tomorrow-
      tide_dur_tomorrow/2;
87
88          float i_tide_amp_today =
      Tide_Amp_Data.getFloat(today_used_calcs,2);
89          float ii_tide_amp_today =
      Tide_Amp_Data.getFloat(today_used_calcs,5);
90
91          if(i_tide_amp_today<=ii_tide_amp_today){
92            l_tide_amp_today = float(Tide_Amp_Data.getInt(today_used_c
      alcs,2));
93            h_tide_amp_today = Tide_Amp_Data.getFloat(today_used_calcs
      ,5);
94            l_tide_time_today = Tide_Amp_Data.getFloat(today_used_calc
      s,1);
95            h_tide_time_today = Tide_Amp_Data.getFloat(today_used_calc
      s,4);
96            l_tide_dur_today = Tide_Amp_Data.getFloat(today_used_calcs
      ,3);
97            h_tide_dur_today = Tide_Amp_Data.getFloat(today_used_calcs
      ,6);
98          }else{
99            l_tide_amp_today = float(Tide_Amp_Data.getInt(today_used_c
      alcs,5));
100           h_tide_amp_today = Tide_Amp_Data.getFloat(today_used_calcs
      ,2);
101           l_tide_time_today = Tide_Amp_Data.getFloat(today_used_calc
      s,4);
102           h_tide_time_today = Tide_Amp_Data.getFloat(today_used_calc
      s,1);
103           l_tide_dur_today = Tide_Amp_Data.getFloat(today_used_calcs
      ,6);
104           h_tide_dur_today = Tide_Amp_Data.getFloat(today_used_calcs
      ,3);
105         }
106
107       h_tide_start = h_tide_time_today-h_tide_dur_today/2;
108         h_tide_end = h_tide_time_today+h_tide_dur_today/2;
109         l_tide_start = l_tide_time_today-l_tide_dur_today/2;
110         l_tide_end = l_tide_time_today+l_tide_dur_today/2;
111
112     for(int i=0;i<5760;i=i+1){
113         if(i>=4*60*(h_tide_time_today-h_tide_dur_today/2) &&
      i<=4*60*(h_tide_time_today+h_tide_dur_today/2)){
114           Tide_Records.setFloat(i ,3 , h_tide_amp_today);
115         }
116         if(i>=4*60*(l_tide_time_today-l_tide_dur_today/2) &&
      i<=4*60*(l_tide_time_today+l_tide_dur_today/2)){
117           Tide_Records.setFloat(i ,3 , l_tide_amp_today);
118         }
119         if(i<4*60*(l_tide_time_today-l_tide_dur_today/2) &&
      i<4*60*(l_tide_time_today-l_tide_dur_today/2)){
120           Tide_Records.setFloat(i ,3 , map(i,-4*60*(24-
      tide_end_yesterday),4*60*(min(h_tide_start,l_tide_start)),tide_amp
      _yesterday,l_tide_amp_today));
```

```
121         }
122         if(i>4*60*(l_tide_time_today+l_tide_dur_today/2) &&
    i>4*60*(h_tide_time_today+h_tide_dur_today/2)){
123           Tide_Records.setFloat(i ,3 ,
    map(i,4*60*max(h_tide_end,l_tide_end),4*60*(24+tide_start_tomorrow
    ),h_tide_amp_today,tide_amp_tomorrow));
124         }
125         if(i>4*60*(l_tide_time_today+l_tide_dur_today/2) &&
    i<4*60*(h_tide_time_today-h_tide_dur_today/2)){
126           Tide_Records.setFloat(i ,3 ,
    map(i,4*60*min(h_tide_end,l_tide_end),4*60*max(h_tide_start,l_tide
    _start),l_tide_amp_today,h_tide_amp_today));
127         }
128
129       }
130       saveTable(Tide_Records,
    "data/Mesocosm_data_for_"+Date_Text+"_"+nf(year(),4)+".csv");
131     }
132     if (create==1){
133       println("new table not needed");
134     }
135
136       cursor(ARROW);
137   }
```

VITA

Daniel Alt graduated from Louisiana State University in December 2015 with a Master of Science in Civil and Environmental Engineering. He has done volunteer work in the United States, Honduras and the Republic of Georgia, and hopes to continue his humanitarian work.